从数据仓库到数据湖

浅谈数据架构演进

Author	Taosheng Shi
WeChat Contact	data-lake
Mail Contact	tshshi@126.com
Organization	NOKIA
Document category	Distributed System
Document location	https://github.com/stone-note/articles

Version	Status	Date	Author	Description of changes
0.1	Draft	12/7/2017	Taosheng Shi	Initiate
0.2	Draft	DD-MM-YYYY	YourNameHere	TypeYourCommentsHere
1.0	Approved	DD-MM-YYYY	YourNameHere	TypeYourCommentsHere

Contents

1 引言 3

2 数据仓库历史沿革 3

3 数据仓库概念 3

4 数据仓库架构 5

5 数据立方体 7

6 数据库建模 9

7 大数据架构 11

8 数据湖架构 18

9 电信运营大数据特点 22

10 演进路径实践 24

1 引言

网管产品需要从数据仓库的角度来看，才能获得完整的视图。数据集成真正从大数据的角度来看，才能明白其中的挑战。一个运行了 20 多年的数据架构，必然有其合理性。也正是因为年代久远，存量过多，才导致举步维艰。在 Cloud 和 5G 时代，超密度网络集成和大数据洞察需求给电信供应商带来新的挑战，从数据仓库到数据湖，不仅仅架构的变革，更是思维方式的升级。本文尝试梳理数据架构的演进过程。

2 数据仓库历史沿革

1970 年，关系数据库的研究原型 System R 和 INGRES 开始出现，这两个系统的设计目标都是面向 on-line transaction processing (OLTP)的应用。关系数据库的真正可用产品直到 1980 年才出现，分别是 DB2 和 INGRES。其他的数据库，包括 Sybase, Oracle, 和 Informix 都遵从了相同的数据库基本模型。关系数据库的特点是按照行存储关系表，使用 B 树或衍生的树结构作为索引和基于代价的优化器，提供 ACID 的属性保证。

到 1990 年，一个新的趋势开始出现：企业为了商业智能的目的，需要把多个操作数据库中数据收集到一个数据仓库中。尽管投资巨大且功能有限，投资数据仓库的企业还是获得了不错的投资回报率。从此，数据仓库开始支撑各大企业的商业决策过程。数据仓库的关键技术包括数据建模，ETL 技术，OLAP 技术和报表技术等。

目前主要的数据仓库产品供应商包括 Oracle、IBM、Microsoft、SAS、Teradata、Sybase、Business Objects(已被 SAP 收购)等。

电信行业是最早采用数据仓库技术的行业之一。由于电信公司运行在一个快速变化和高速竞争的环境，拥有大量的客户基础，从而产生和存储海量的高质量数据。电信公司利用数据挖掘技术降低营销成本，识别欺诈，并更好地管理其电信网络。

3 数据仓库概念

数据仓库之父 Bill Inmon 在 1991 年出版的“Building the Data Warehouse”一书中所提出的定义被广泛接受——数据仓库（Data Warehouse）是一个面向主题的（Subject Oriented）、集成的（Integrated）、相对稳定的（Non-Volatile）、反映历史变化（Time Variant）的数据集合，用于支

持管理决策(Decision Making Support)。这是一个偏向学术的定义，却非常准确的界定了数据仓库与其他数据库系统的本质区别。

“A data warehouseis a subject-oriented, integrated, time-variant, and nonvolatile collection ofdata in support of management’s decision-making process.”

—W. H. Inmon

要理解数据仓库的概念，需要从与数据库的系统的对比来看。

数据库是作为“所有处理的单一数据源”出现和定义的。数据库的出现有两个驱动因素，第一是 70 年代以前大量应用程序和主文件的分散存放导致一片混乱和大量冗余数据。第二是直接存取存储设备的出现使得按记录寻址成为可能。基于 DBMS 的在线事务处理为商业发展开辟全新的视野。

数据库系统的设计目标是事务处理。数据库系统是为记录更新和事务处理而设计，数据的访问的特点是基于主键，大量原子，隔离的小事务，并发和可恢复是关键属性，最大事务吞吐量是关键指标，因此数据库的设计都反映了这些需求。

数据仓库的设计目标是决策支持。历史的，摘要的，聚合的数据比原始的记录重要的多。查询负载主要集中在即席查询和包含连接，聚合等操作的复杂查询。相对于数据库系统来说，查询吞吐量和响应时间比事务处理吞吐量重要的多。

数据仓库和数据库系统的区别，一言蔽之：OLAP 和 OLTP 的区别。数据库支持是 OLTP，数据仓库支持的是 OLAP。

数据处理类型	OLTP	OLAP
User orientation	业务开发人员	分析决策人员
System orientation	Customer-oriented	Market-oriented
功能实现	日常事务处理	面向分析决策
数据模型	关系模型(ER)、面向应用	多维模型（星型或雪花）、面向主题
数据量	几条或几十条记录	百万千万条记录
Data contents	Current data	Historic data
View	单个企业的当前数据	多个企业的历史数据
操作类型	短并发事务：查询、插入、更新、删除	查询为主：只读操作、复杂查询

对 OLTP 和 OLAP 的区别还可以有一个维度，就是及时性需求。OLTP 对事务的及时性需求较高，而 OLAP 则不然。

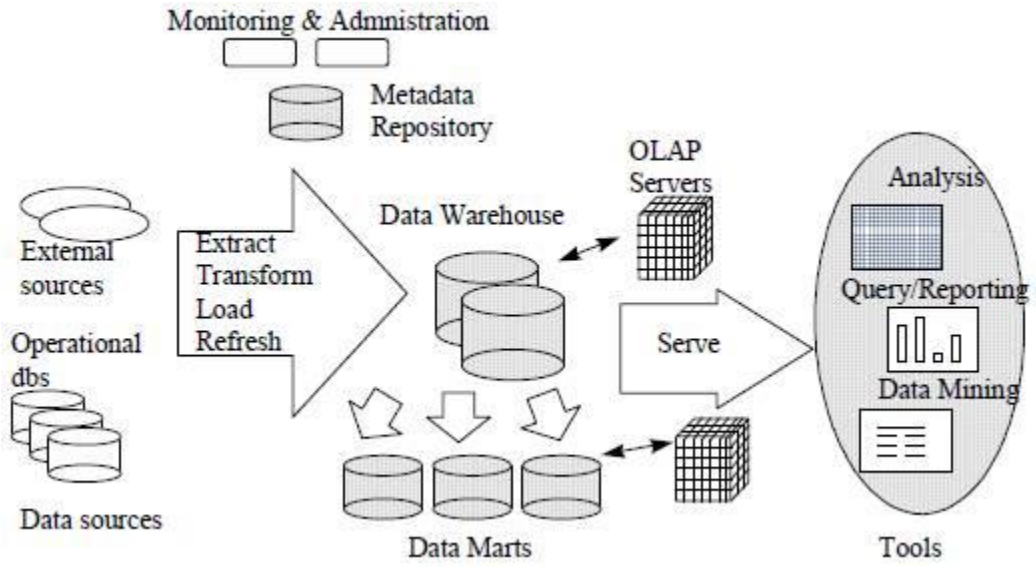
——曹洪伟

数据仓库一般基于数据库实现，但是为部署和维护上是分离的。数据仓库可以是基于关系数据库实现的，这样的数据仓库被称为 ROLAP。数据仓库也可以是基于多维数据结构实现的，这样的数据仓库被称为 MOLAP。

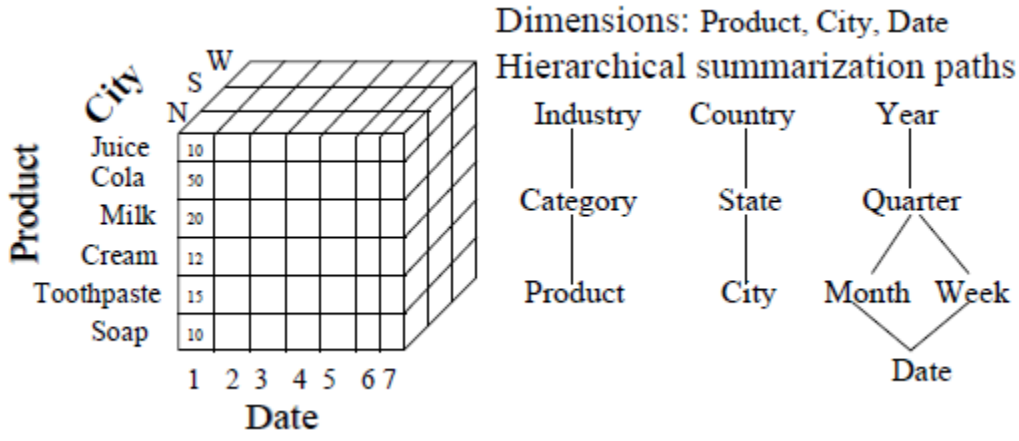
4 数据仓库架构

数据仓库是一种体系结构，而不是一种技术。数据仓库最为核心的内容分类两部分：

- 基于关系数据库的多维建模（RDBMS-based dimensional modeling）
- 基于数据立方体的 OLAP 查询（cube-based OLAP）



数据仓库体系结构包含了从外部数据源或者数据库抽取数据的 ETL 工具。ETL 还负责数据的转换，清洗，然后加载到数据仓库的存储中。一般来说，数据都会加载到存取速度较慢的存储中，以原始数据的方式保存下来。为了提高查询效率，原始数据会按主题分类，以聚合的方式存储到数据集市中，称之为聚合数据。参见下图，原始数据往往有多条聚合路径，时间维度是一个最基本的内置聚合路径，行政级别划分也是一种常见的聚合路径，产品属性也是常见的聚合路径。



数据仓库体系结构中还包括前端的查询工具，报表工具和数据挖掘工具，被称为 front-end。

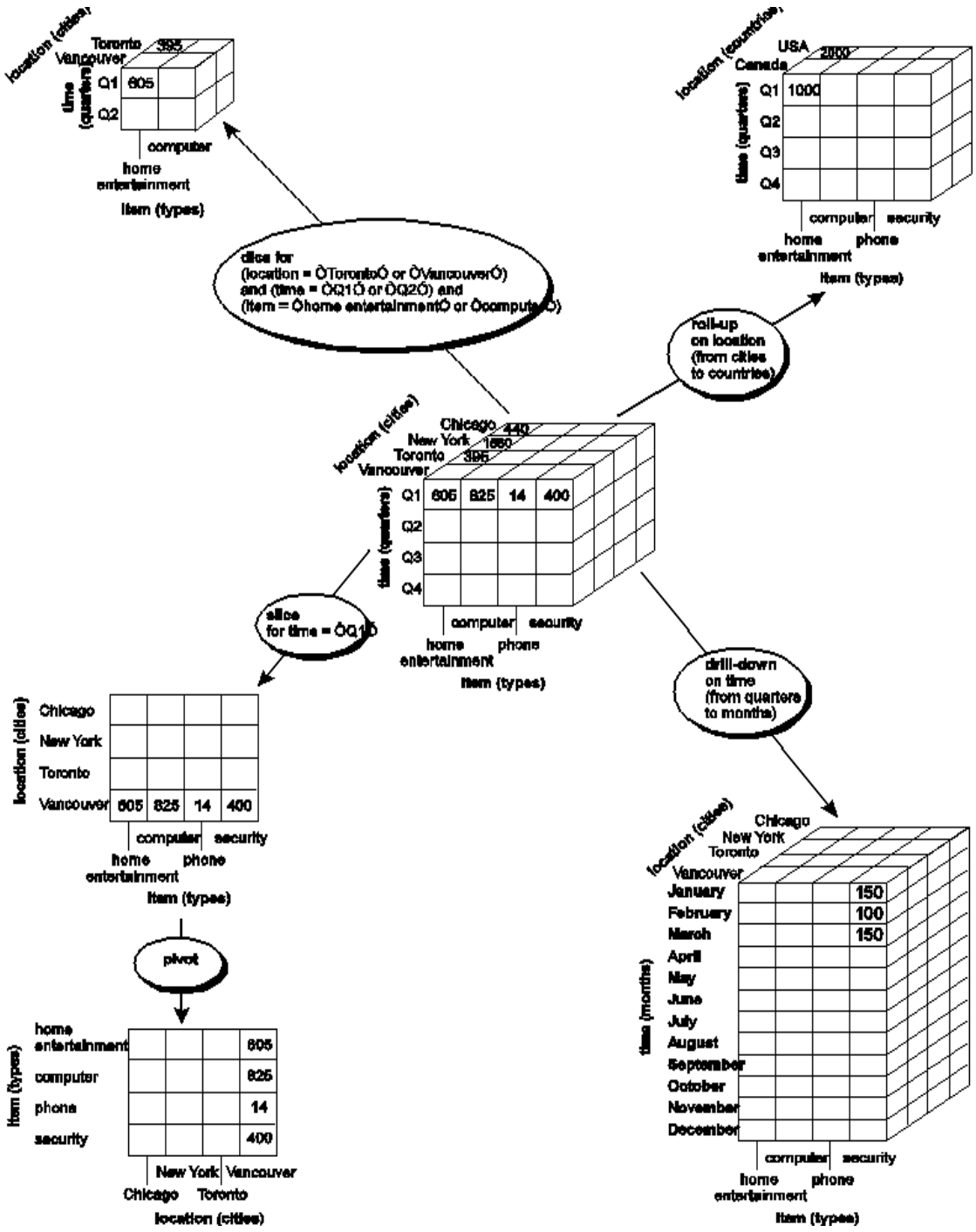
最后也是最重要的是，数据仓库体系结构中都会包含一个构建数据仓库的元数据仓库。元数据仓库包括数据库 schema，view，用于 ETL 的 metadata，用于数据聚合的 metadata，用于报表呈现的 metadata 和 SQL 模板等。数据仓库往往采用 meta data driven 的架构设计，这个元数据仓库就至关重要。

上文中提到的维度的概念。维度(dimension)是观察事物的角度，也是数据库事实表中用来描述数据分类的层次结构。维度在数据中就是表示为列，在 SQL 中用作过滤和分组。

像上图这样对数据进行多个维度的抽象并借助于数据库的 select，group by 等基本操作形成的 OLAP 多维数据操作（roll up，drill down，slice and dice，pivot）被称为多维数据模型。为了方便复杂分析和可视化呈现，数据仓库中数据往往以多维模型建模。每一个维度被称为一个层级，三个维度构成一个数据立方体。维度也通常用来过滤和分组，所以数据立方体称之为 group by 的并。OLAP 也被称为在基于数据仓库多维模型的基础上实现的面向分析的各类操作的集合。

5 数据立方体

数据立方体只是多维模型的一个形象的说法。立方体其本身只有三维，但多维模型不仅限于三维模型，可以组合更多的维度，但一方面是出于更方便地解释和描述，同时也是给思维成像和想象的空间；另一方面是为了与传统关系型数据库的二维表区别开来，于是就有了数据立方体的称呼(见下图)。



OLAP 的操作是以查询——也就是数据库的 SELECT 操作为主，但是查询可以很复杂，比如基于关系数据库的查询可以多表关联，可以使用 COUNT、SUM、AVG 等聚合函数。OLAP 的多维分析操作包括：钻取（Drill-down）、上卷（Roll-up）、切片（Slice）、切块（Dice）以及旋转（Pivot），逐一解释如下：

- Roll up (drill-up): summarize data by climbing up hierarchy or by dimension reduction
- Drill down (roll down): reverse of roll-up,from higher level summary to lower level summary or detailed data, or introducing new dimensions

- Slice and dice: project and select
- Pivot (rotate): reorient the cube, visualization, 3D to series of 2D planes

看了上图中数据立方体的各种操作，有人觉得还是很抽象。下面给一个 SQL 的例子，说明数据立方体的具体操作。

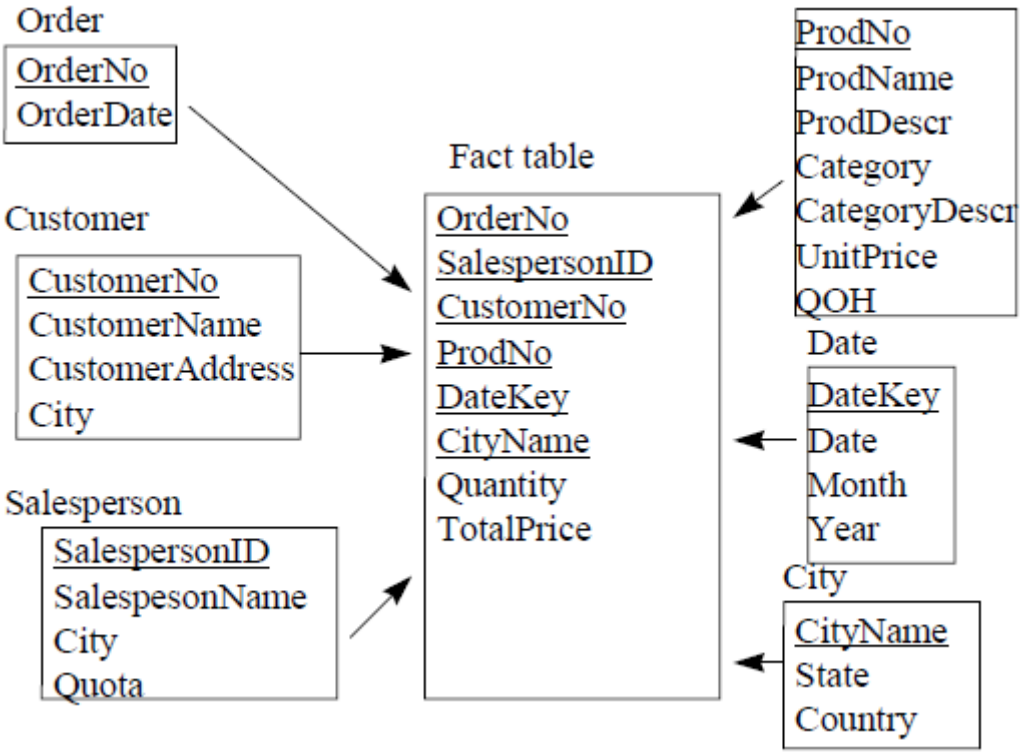
```
select//公式必须配合 group by 使用
   tmp.time,
   tmp.id1,
   tmp.id2,
   SUM(counter1) counter1,
   SUM(counter2) counter2
from//双层 SQL，实现聚合路径
   (
      select//trunc 实现时间维度的变化
            trunc( p.time, 'min' ) time,
            "country".country_id id1,
            "city".city_id id2,
            SUM(p.counter1) counter1,
            SUM(p.counter2) counter2
      from
         table "country"，
         table "city"，
         table p
      where//选择计算的城市
            "city".city_id in ( '北京','上海','广州' )
            and time >= to_date('2016/01/01 00:00:00', 'yyyy/mm/dd hh24:mi:ss')
            and time < to_date('2017/01/01 00:00:00', 'yyyy/mm/dd hh24:mi:ss')
      group by//改变行政维度
            trunc( p.period_start_time, 'mi' ),
            "country".country_id,
            "city".city_id
   ) tmp
group by//行政维度可以不变
   tmp.time,
   tmp.id1,
   tmp.id2
```

OLAP 的优势是基于数据仓库面向主题、集成的、保留历史及不可变更的数据存储，以及多维模型多视角多层次的数据组织形式，如果脱离的这两点，OLAP 将不复存在，也就没有优势可言。基于多维模型的数据组织让数据的展示更加直观，它就像是我们平常看待各种事物的方式，可以从多个角度多个层面去发现事物的不同特性，而 OLAP 正是将这种寻常的思维模型应用到了数据分析上。

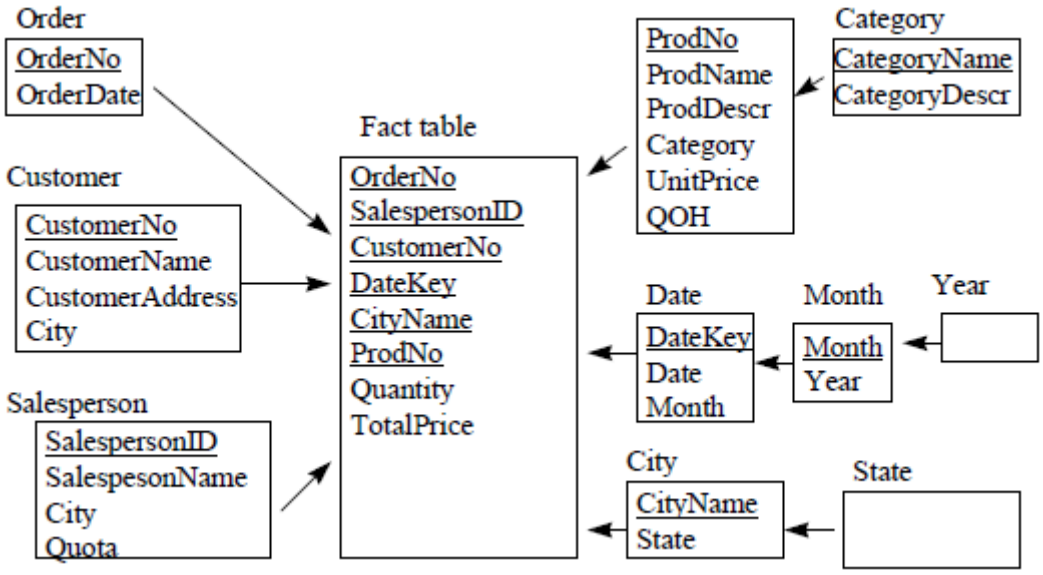
6 数据库建模

如果把多维数据模型映射到关系数据库和 SQL 查询上（ROLAP），数据库该如何设计呢？

大多数数据仓库都采用“星型模型”来表示多维数据模型。在星型模型中，只有一个事实表，并且每一个维度有一个单独的表。事实表中的每一个元组都是一个外键指向维度表的主键。每一个维度表的列是组成这个维度的所有属性。如下图所示。



另外一个常见的数据库设计方法是“雪花模型”。雪花模型通过定义单独的维度表，改进了星型模型中没有明确提供维度层级的问题。是谓维度表的正则化，如下图。但星型模型更适合浏览维度层级。

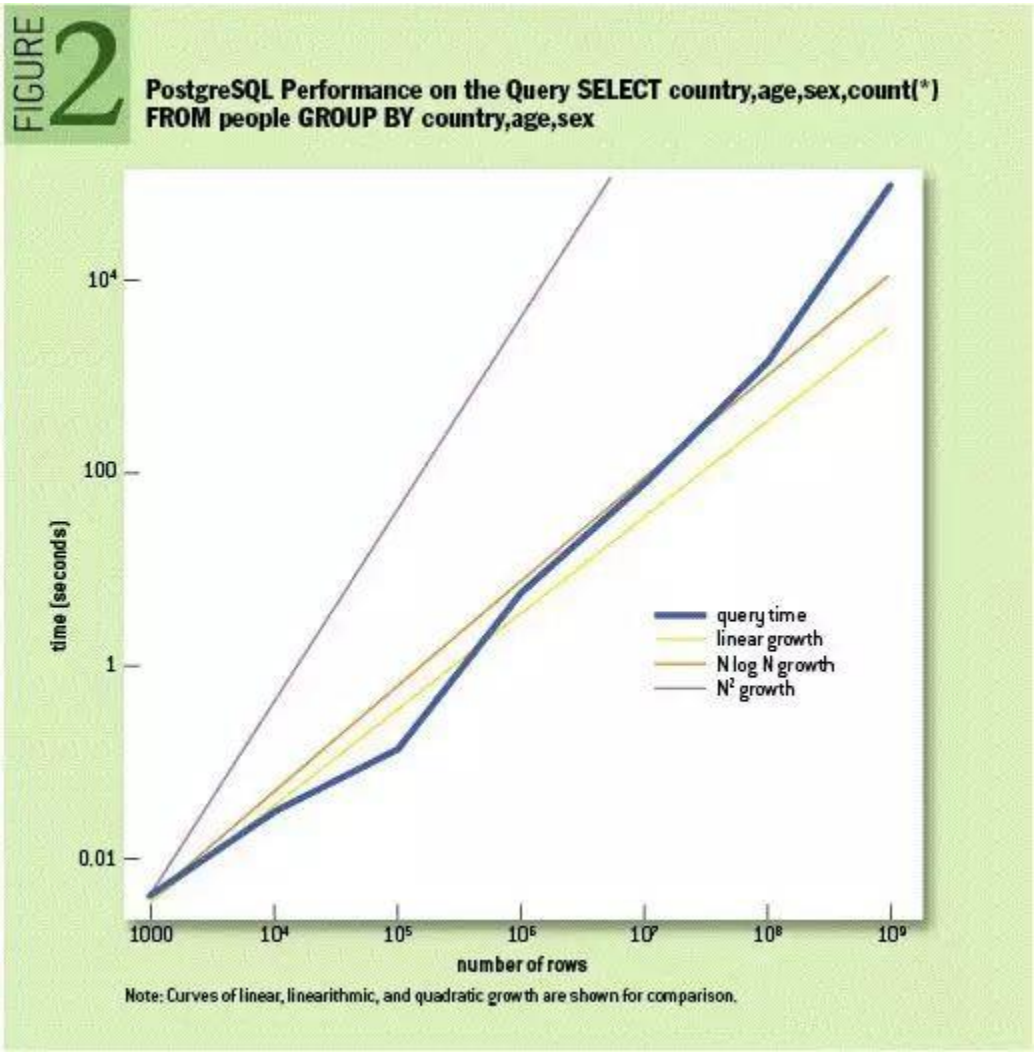


除了事实表和维度表，数据仓库还需要创建 pre-aggregation 表用于存储挑选的摘要数据。

7 大数据架构

1010data 公司高级软件工程师 ADAM JACOBS 博士在 ACM 通讯发表的《大数据病理学》指出大数据的病理在于分析而不在于存储——我们期望从成年累月积累的数据中在几分钟或者几秒内获得分析结果！其实作者指出了关系数据库的在大数据时代的病理，如下图所示一个数据仓库分析操作的 SQL 在数据量超过 100 万条记录时的性能表现。

The pathologies of big data are primarily those of analysis.



因此，数据仓库被认为是对数据库查询性能问题的一个解决方案。在 90 年代，人们已经都面临一个数据爆炸的挑战，为了解决那个时代的“大数据”问题，数据仓库应运而生。

- *在 1980s 早期，大数据是指数据集超出了磁带机的处理能力。*
- *在 1990s，大数据是指数据集超出了 Microsoft Excel 或者桌面 PC 的处理能力。*
- *今天，大数据是指数据集超出了关系数据库的处理能力。*

站在大数据时代回望数据架构的发展历史，然后从技术的角度思考大数据的定义：

当前流行的技术处理不了的数据，都是大数据。

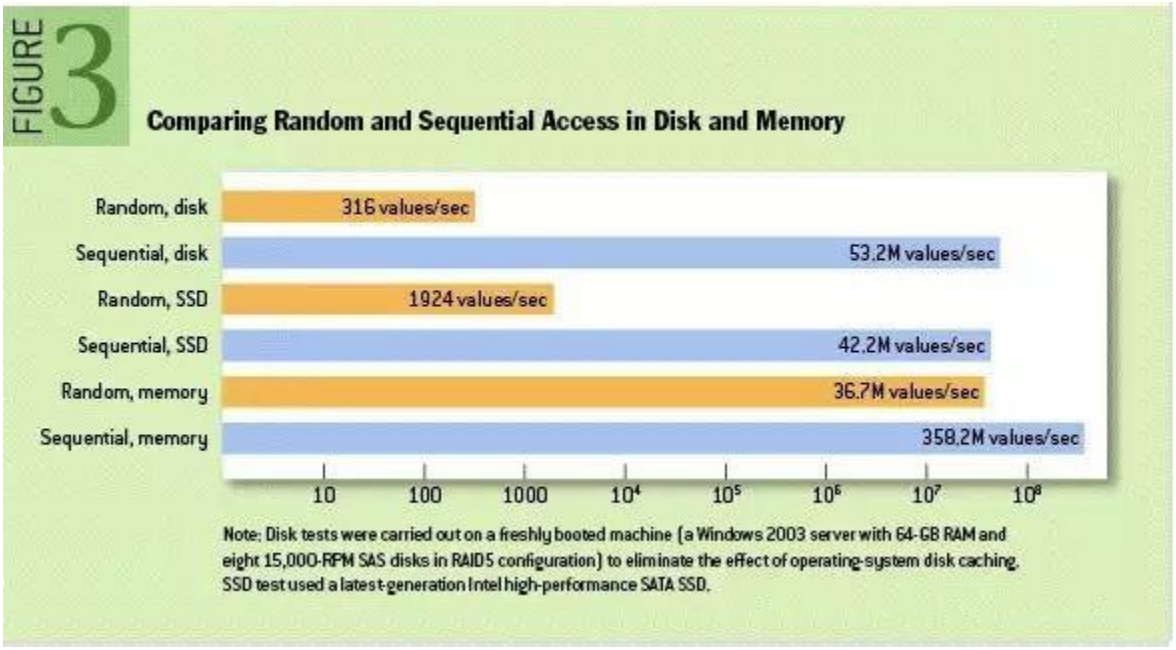
数据仓库的本质是把数据变小，一般有两个方法：

- 第一是通过抽取，转换，加载，清洗。
- 第二是通过 pre-aggregation 获得数据的一份单独拷贝。因此数据仓库被定义为：

为了方便查询分析，把数据从关系数据库中单独拷贝一份出来，然后通过 ETL 或者 ELT 转换。

对于大数据，仅仅简单构建一个数据仓库是不够的。数据应该如何结构化才能更便于分析？数据库和分析工具应该如何设计才能更高效的处理大数据？

意识到大数据固有的时间属性和空间属性，是我们理解关系数据库处理大数据时存在性能问题的重要前提。如果说数据是我对世界的观察记录的话，大数据是我们对世界在时间和/或空间维度的重复观察。这就是大数据的时空特点，也是数据仓库多维模型的构建原理。当今的主流数据库模型是关系数据库，并且该模型显式地忽略表中的行的顺序。这将不可避免导致应用以非顺序的方式查询数据。在这种情况下，传统的数据架构可以通过引入缓存的方式缓解性能问题，而大数据则会大大放大了次优访问模式对性能的影响。如下图所示随机访问和顺序访问的差别。



因此我们要引入，也是我们要推导的结论：逆正则化（逆规范化）和顺序存储，不可更改数据集（append only，immutable data set）。顺着存储栈往下走，直到数据存储格式。

是时候放弃关系数据库了。

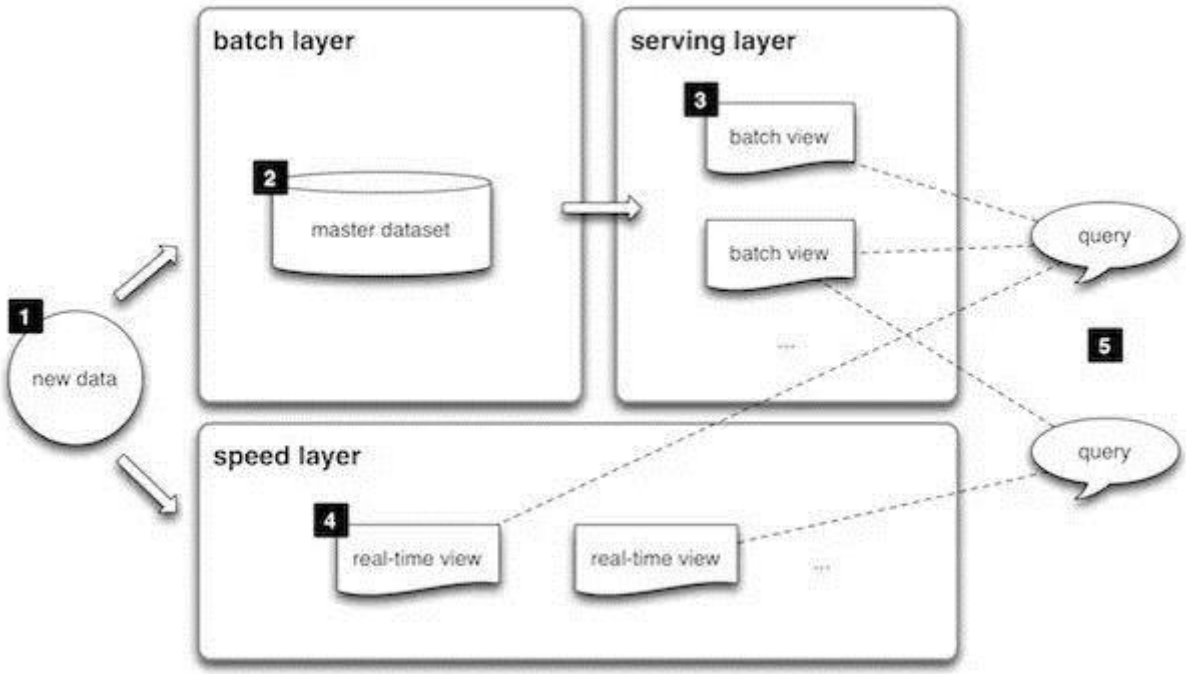
简单解释一下逆正则化（逆规范化）。经典关系数据库介绍的所有范式指导思想都是正则化，减少重复数据，如果重复，则单独创建一个表，使用外键关联，目的是节省存储空间（那个时候存储很昂贵）。逆正则化则是允许列之间的重复。如下图所示。

我有一个看法，NoSQL 的键值存储即是用极简的非结构化来实现结构化存储的逆规范化。

键值存储是极简的结构化，也是极简的非结构化。

关于顺序存储，不可更改数据集，可以参考 Pat Helland《Immutability Changes Everything》，和我上面的介绍是一致的。

关于传统关系数据库的讨论还有数据库知名专家，2015 年图灵奖得主 Michael Stonebraker 撰写的《One Size Fits All》，分别从数据仓库和流处理两个方面探讨了数据库 25 年来一招不变的灵丹妙药已经不再适合现在的业务发展。文章的中心思想和 Pat Helland 提出 lambda 架构也有异曲同工之妙。



speed layer

(i) compensates for the high latency of updates to the serving layer

(ii) deals with recent data only

serving layer

(i) indexes the batch views

(ii) Can be queried in low-latency, ad-hoc way

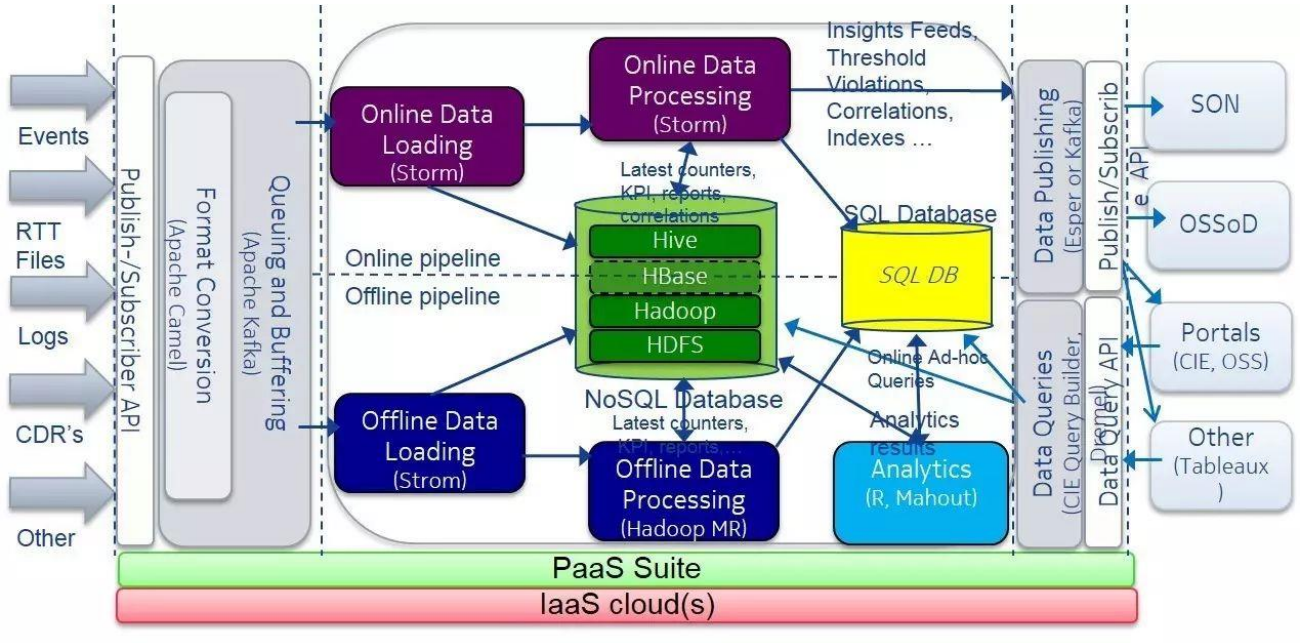
batch layer

(i) managing the master dataset (an immutable, append-only set of raw data),

(ii) pre-compute the batch views

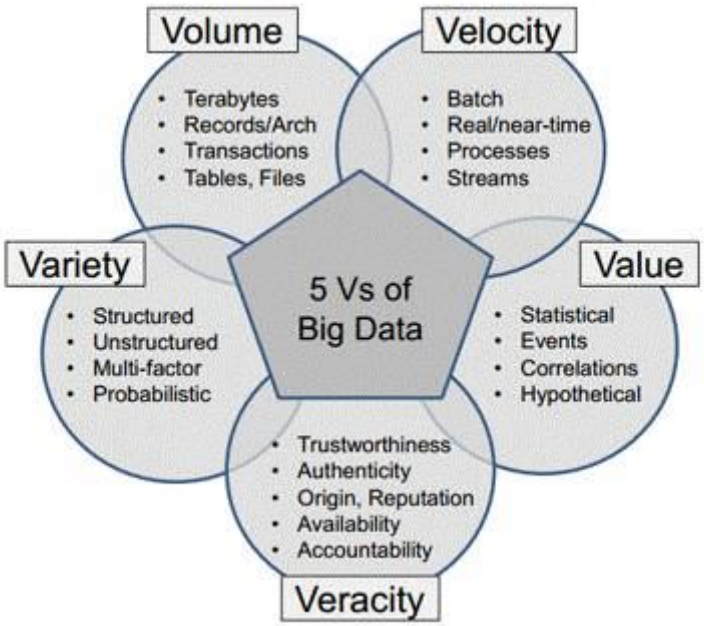
Lambda 架构统一了传统数据仓库时代的半实时在线查询，刚刚兴起的实时流处理（Online ），和批处理数据分析（Offline），给数据架构的设计人员提供了一个全面的参考。

再结合半结构化，结构化数据存储，SQL and No-SQL 混合，我们可以得到下面一个典型的数据架构：



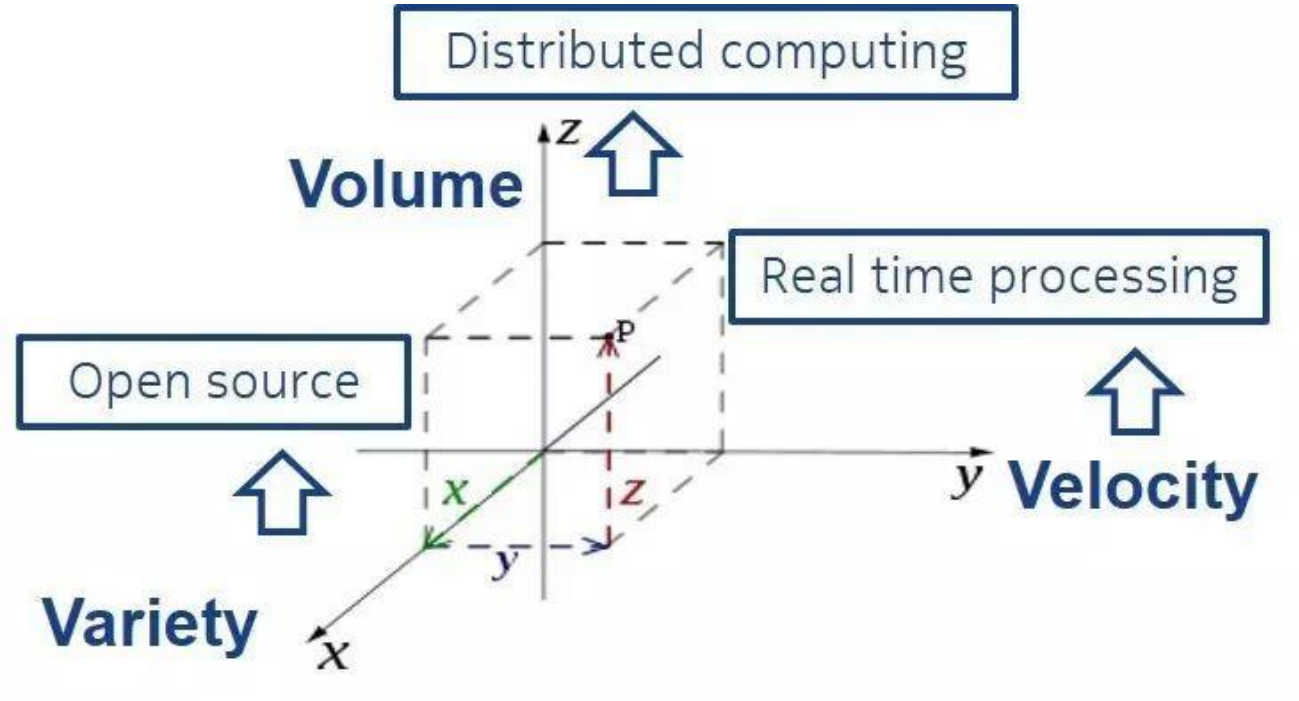
上面的讨论是架构的微观考虑，让我们回到大数据架构的宏观指导上来。

目前业界对大数据的一个共识的定义是 5 个 V。如下图所示。



从技术的角度需要专注于其中的三个 V，通过阅读大量文献，我得到下面一个范型：

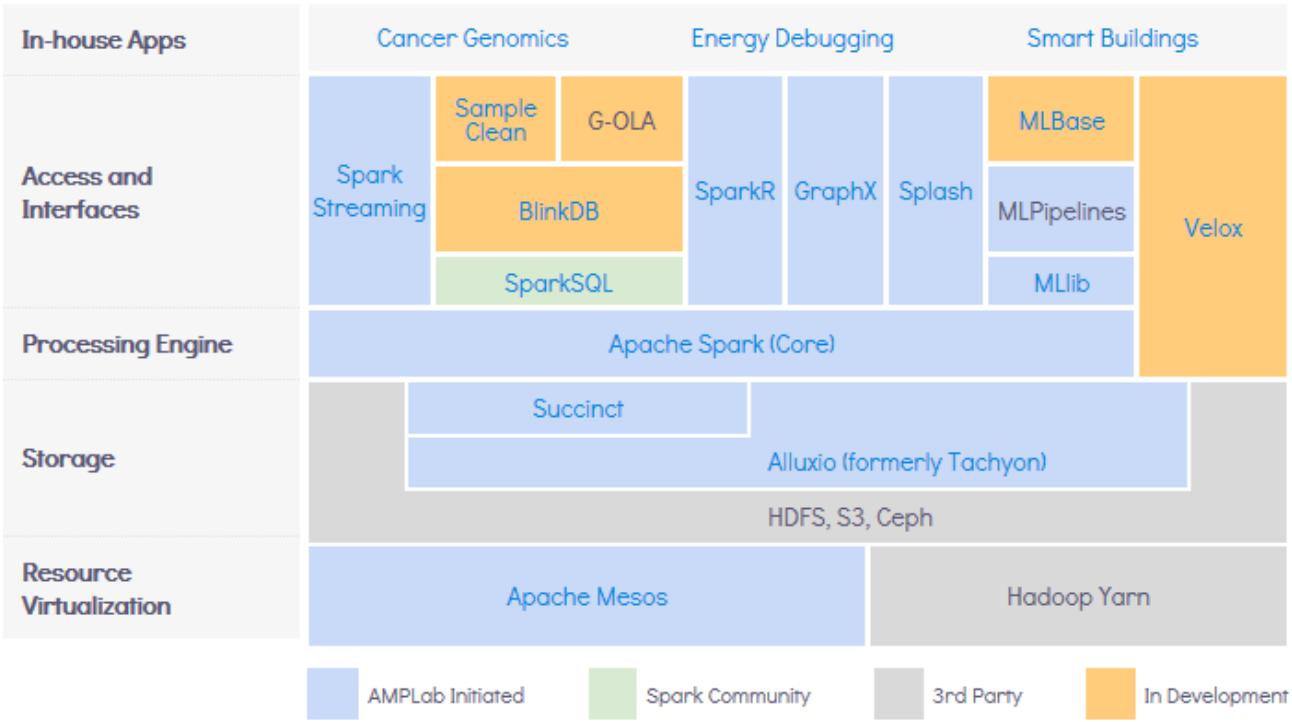
- 借力开源软件处理数据多样性挑战
- 使用分布式技术解决数据容量问题
- 使用实时流处理技术解决数据速度问题



传统的 OLAP 而言，实时性需求不明显，实时分析的强需求是导致大数据技术的一个原因。

——曹洪伟

基于此，我个人推荐的大数据架构是 BDAS, the Berkeley Data Analytics Stack。这个架构中不仅包含上面提到的三个思考维度，还提供了整个大数据架构 blueprint。内容很多，使用时各个击破，在此不赘述。



谈了那么多，总结一下大数据架构的几个要点：

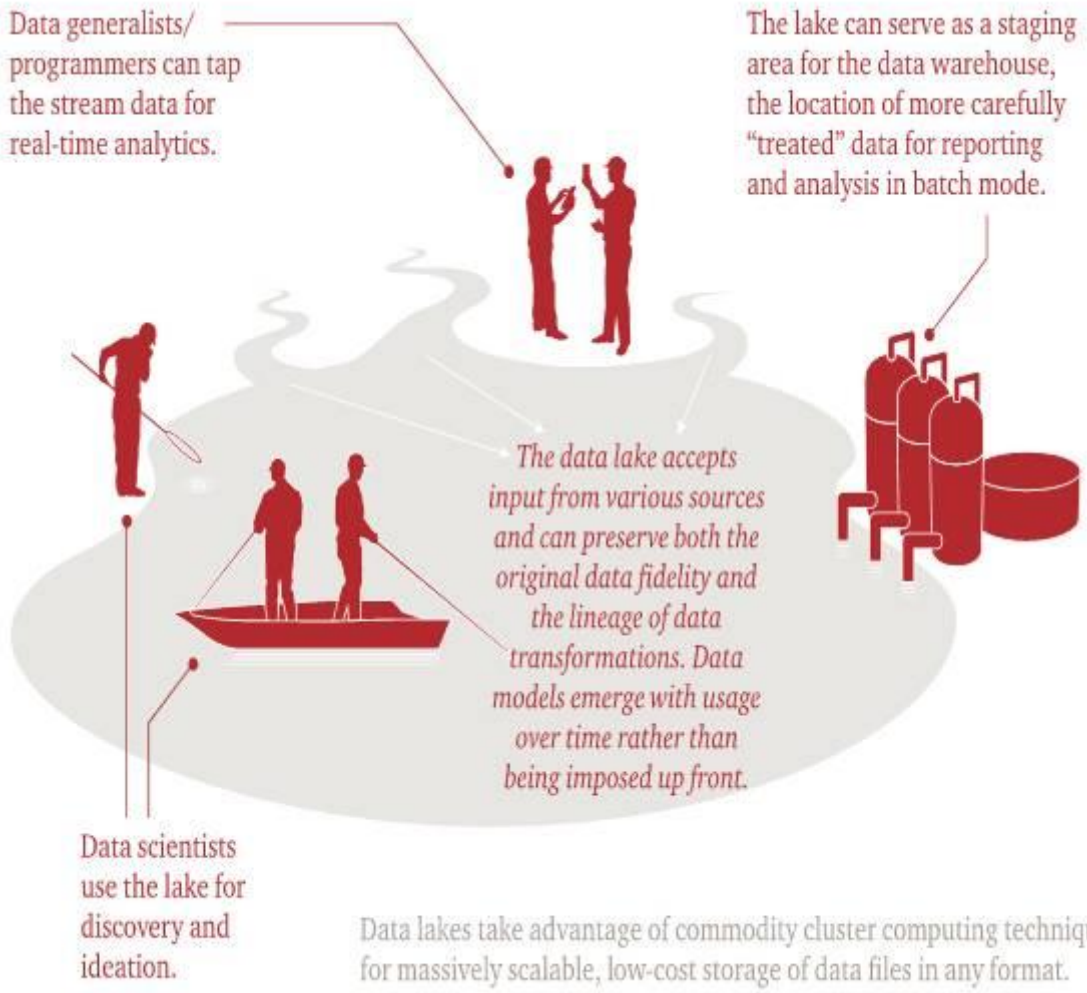
- 分布式计算
- 实时流处理
- Online 和 Offline
- SQL 和 No-SQL：混合架构也是演进路径之一
- 逆正则化（逆规范化）和顺序存储，不可更改数据集

8 数据湖架构

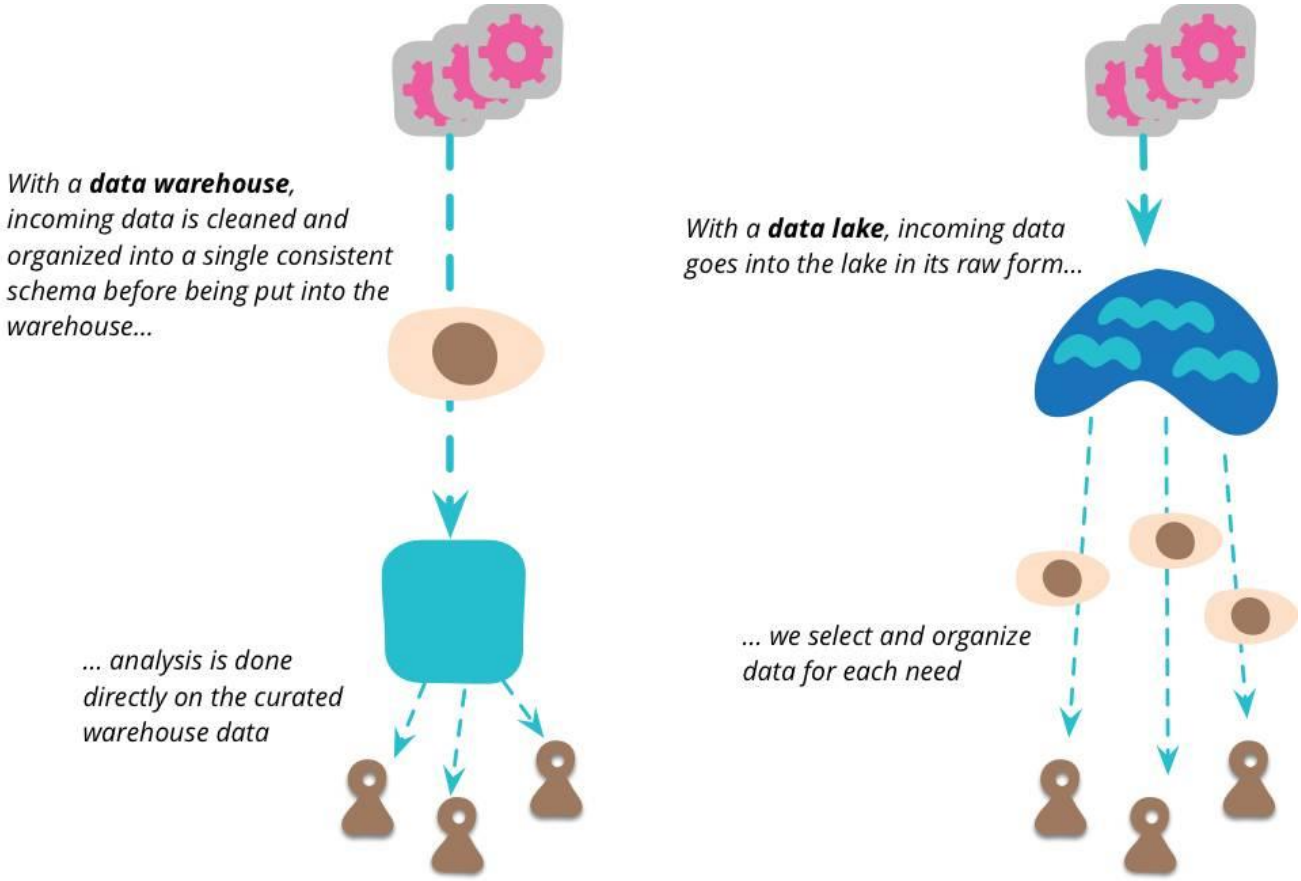
Pentaho 的 CTO James Dixon 在 2011 年提出了“Data Lake”的概念。在面对大数据挑战时，他声称：不要想着数据的“仓库”概念，想想数据 的“湖”概念。数据“仓库”概念和数据湖概念的重大区别是：数据仓库中数据在进入仓库之前需要是事先归类，以便于未来的分析。这在 OLAP 时代很常见，但是对于离线分析却没有任何意义，不如把大量的原始数据线保存下来，而现在廉价的存储提供了这个可能。

Nearly unlimited potential for operational insight and data discovery. As data volumes, data variety, and metadata richness grow, so does the benefit.

形象的来看，如下图所示，数据湖架构保证了多个数据源的集成，并且不限制 schema，保证了数据的精确度。数据湖可以满足实时分析的需要，同时也可以作为数据仓库满足批处理数据挖掘的需要。数据湖还为数据科学家从数据中发现更多的灵感提供了可能。



和数据仓库对比来看，数据仓库是高度结构化的架构，数据在转换之前是无法加载到数据仓库的，用户可以直接获得分析数据。而在数据湖中，数据直接加载到数据湖中，然后根据分析的需要再转换数据。

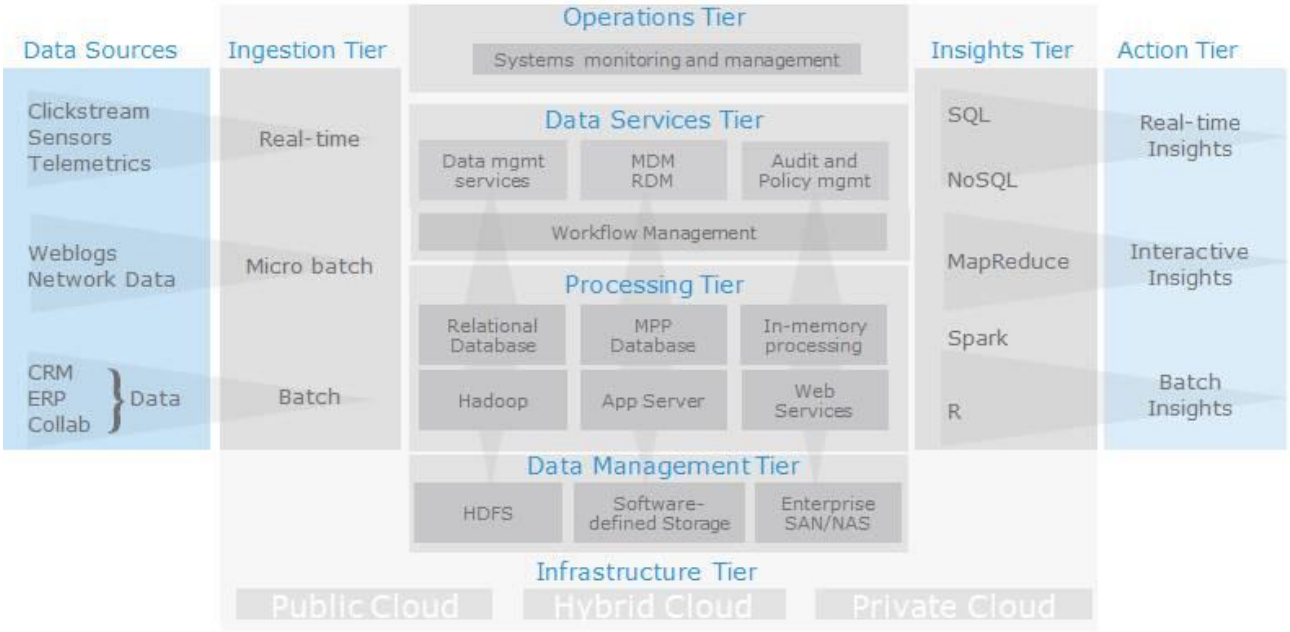


下面我整理了数据仓库和数据湖在多个维度的详细对比。

Dimension	Enterprise Data Warehouse	Data Lake
Schema	Schema on write 数据存储之前需要定义 schema 数据集成之前完成大量工作 数据的价值应提前明确	Schema on read 数据存储之后才需要定义 schema 提供敏捷，简单的数据集成 数据的价值应尚未明确
Scale	中等开销获得较大的容量扩展	低成本开销获得极大容量扩展
Access methods	标准的 SQL 接口或者 BI 接口 ANSI SQL, ACID compliant Seeks	应用程序，类 SQL 的程序，其他方法 Flexible programming, evolving SQL Scans
Data	清洗过的数据 结构化的数据	原始数据 结构化，半结构化数据
Complexity	复杂的 SQL 连接	复杂的大数据处理
Security	成熟的	发展中
Benefit	高并发 快速响应 多数据源集成 干净，安全的数据 转换一次，多次使用	无限扩展性 并行执行 支持编程框架 数据经济

总结起来，数据湖架构有一下几个显著的特点：

- 数据存储：大容量低成本
- 数据保真度：数据湖以原始的格式保存数据
- 数据使用：数据湖中的数据可以方便的被使用
- 延迟绑定：数据湖提供灵活的，面向任务的数据绑定，不需要提前定义数据模型

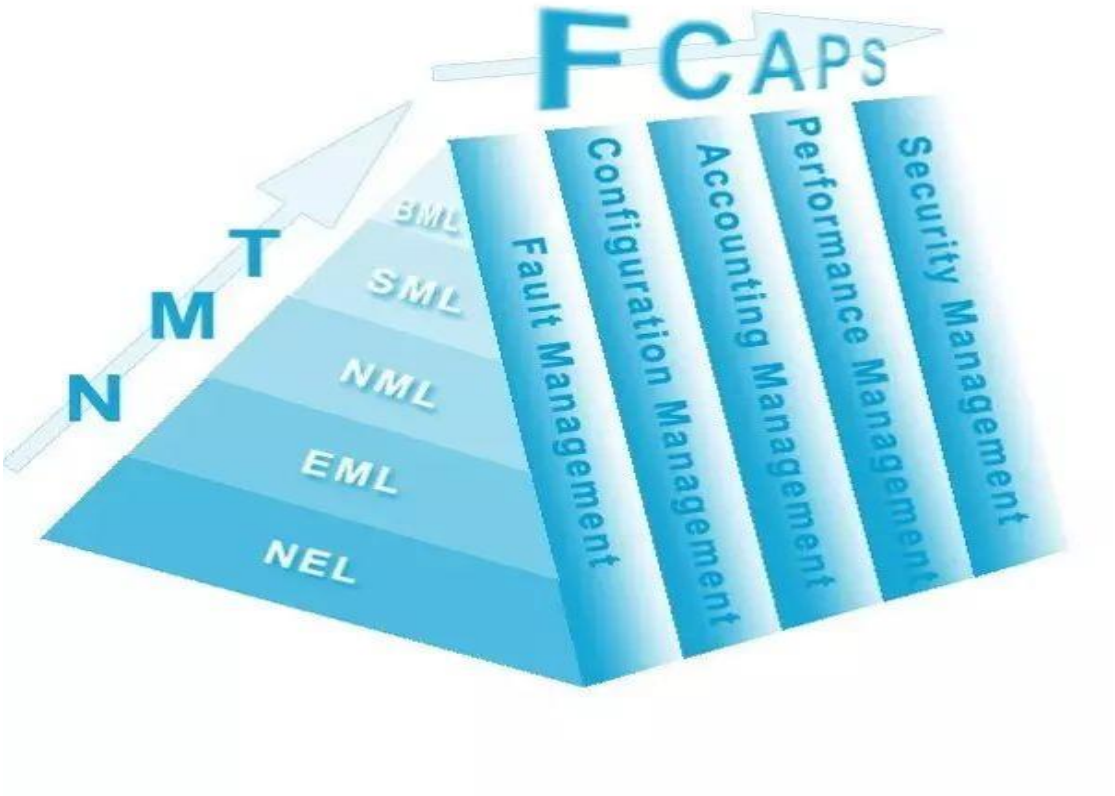


当然，对于数据湖架构的批评也是不绝于耳。有人批评说，汇集各种杂乱的数据，应该就是数据沼泽。Martin Fowler 也对数据湖中数据的安全性和私密性提出了质疑。

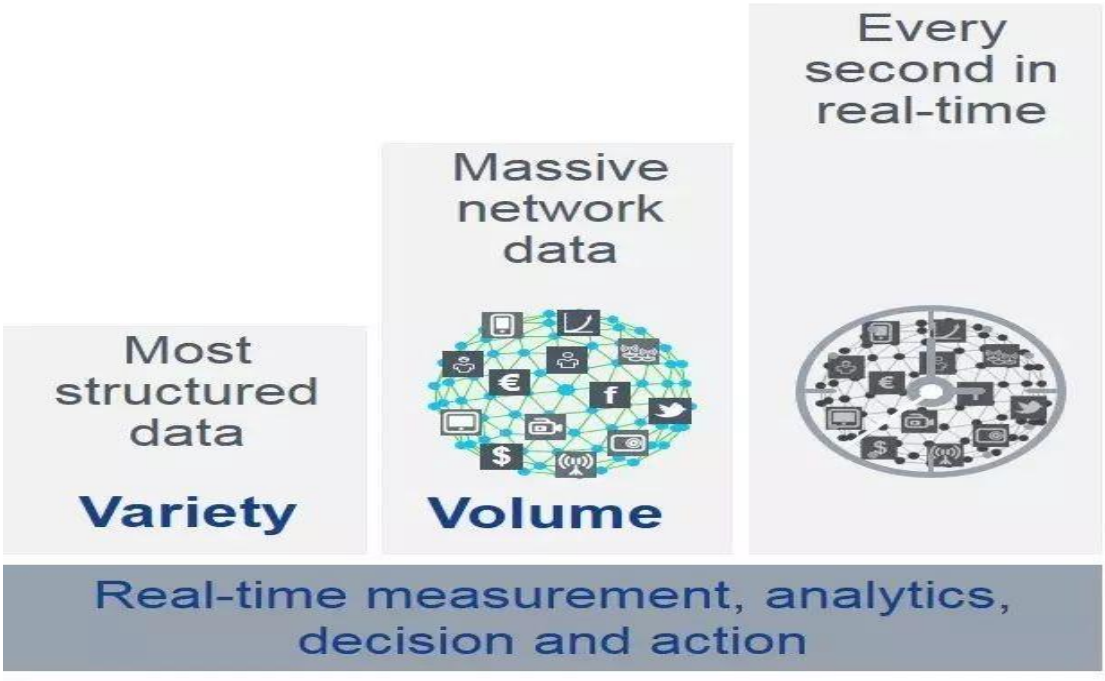
9 电信运营大数据特点

电信运营大数据对应于 TMN/FCAPS 模型中的电信设备管理数据。如下图所示。

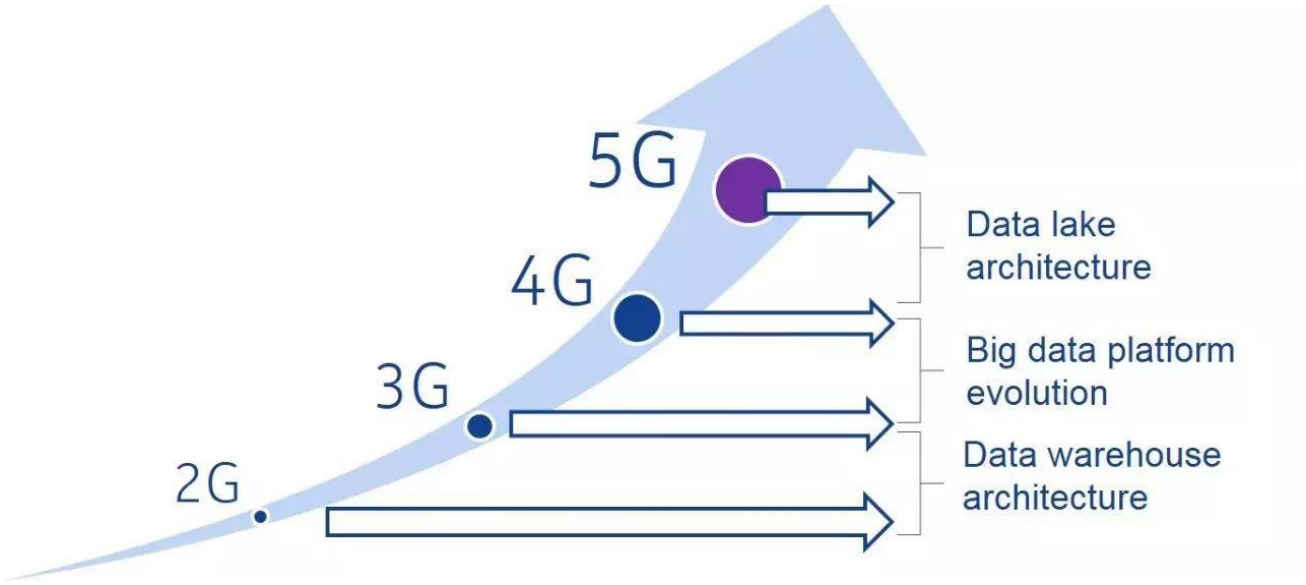
- Fault Management
- Configuration Management
- Accounting Management
- Performance Management
- Security Management



电信运营数据的特点是数据多样化要求不高，大多数数据是结构化数据，数据容量要求不是特别高，数据的实时处理要求最高。



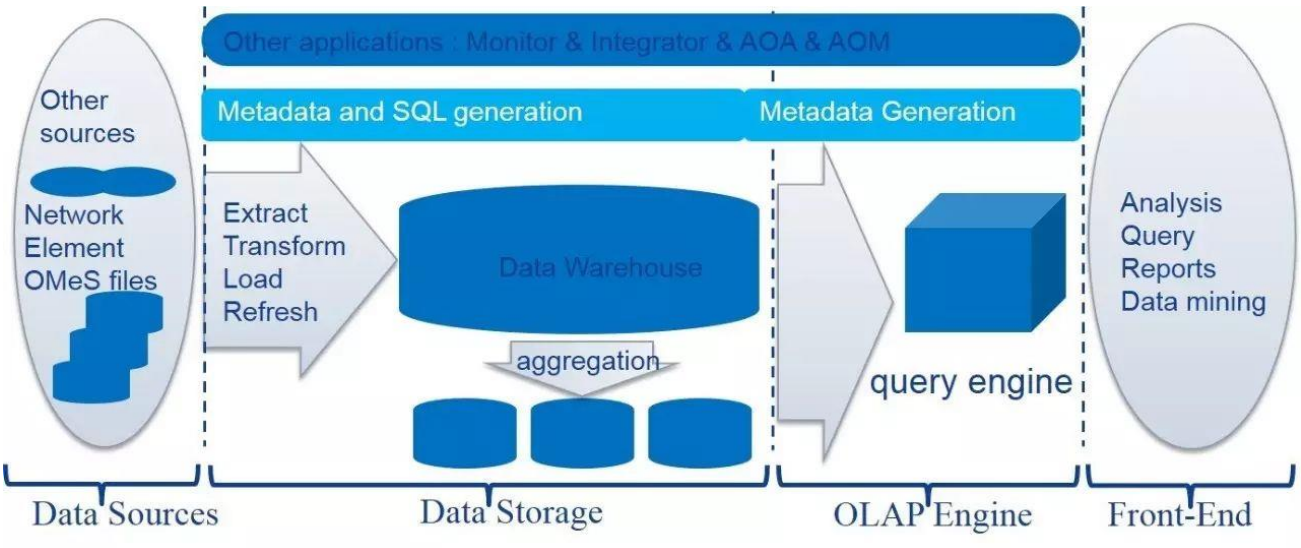
电信运行数据架构强调演进。步步为营，向前兼容，不是一蹴而就的。



10 演进路径实践

现在的架构是一个典型的数据仓库架构。如下图所示。现在的架构设计有以下几个要点：

- ROLAP：基于 Oracle 数据库，但并没有用 Oracle 的数据仓库，单独构建数据仓库。
- Meta Data Driven 的架构设计：Meta Data 覆盖整个数据 pipe。当新的数据需要集成，只需要编辑新的 Meta Data，系统不需要做任何改变。
- Schema 设计：主要有两类表：原始数据表和聚合表； 每类表都有三层结构：表，用作聚合的视图，用作报表的视图。不同的应用使用不同的视图来操作数据。当原始的数据表结构变化时，可以根据需要更改不同层次的视图。
- Schema 的演化。这是一个比较大的主题，关系数据是 schema on write 的，任何列的增加都需要 alter 表结构，这会带来客户系统很长时间的 downtime。因此原始表采用 1000 列的设计（Oracle 支持的最大列数），并且列只增加，不减少，避免了数据库 schema 的变化，降低不同 release 之间 migration 的成本。
- 数据存储：定期清除原始数据，只保留聚合数据。



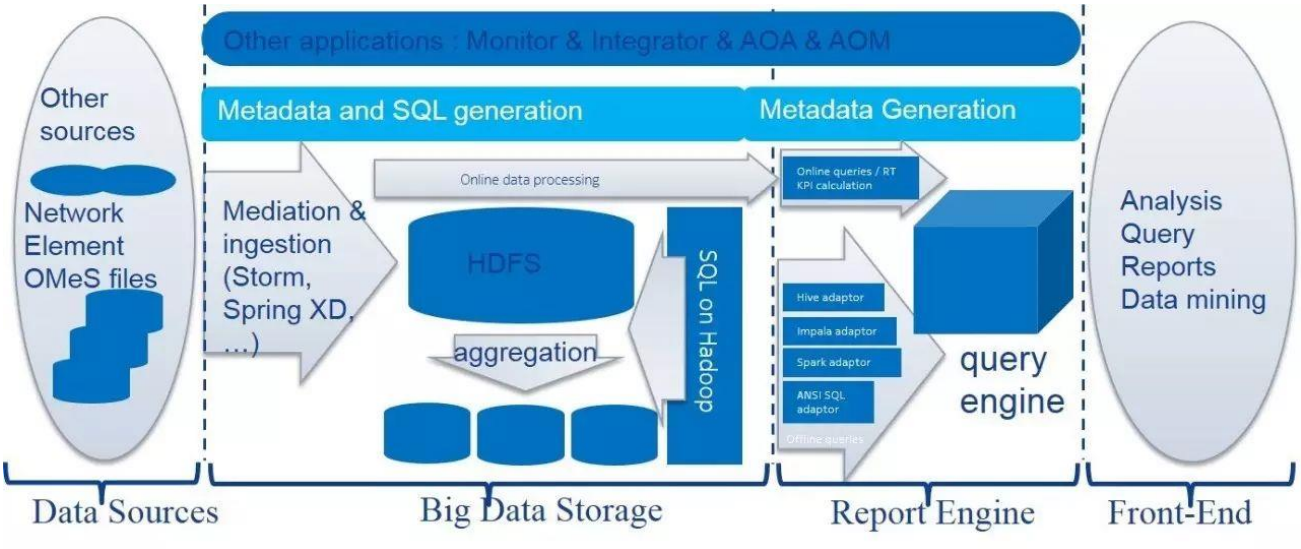
为什么现在的架构需要演进呢？

首先当前架构面临扩展性的挑战。数据库扩展性主要依赖于 Oracle RAC 解决方案，Oracle RAC 不是一个线性的扩展方案，同时也增加了很多管理和维护成本。并且由于硬件的限制，垂直性扩展不是一个长期的解决方案。

其次，当前的存储成本太昂贵，因此去 IOE 成为目标。

第三，实时处理需求也是驱动架构演进的重要因素。

然后，架构变成了这样子：



传统 SQL 基于云平台重新定义为 NewSQL，那么 Data Warehouse 也可以重新定义 New Data Warehouse。

——曹洪伟

这样的架构是不是 New Data Warehouse，我不知道，可能是。在这样的架构下，最大的变化就是更换 Oracle 数据为 HDFS，并使用 SQL on Hadoop（比如 Hive SQL，Spark SQL）等保持 SQL 接口，维持了前端分析引擎的不变。Meta Data 部分依然保持了原来的数据建模，并没有改变数据集成方式。这样的架构继承了经典的仓库架构，提高系统扩展性，在满足业务需求的同时，最大化的保护已有投资。

在架构演进这个过程中，有一些 lesson learned：

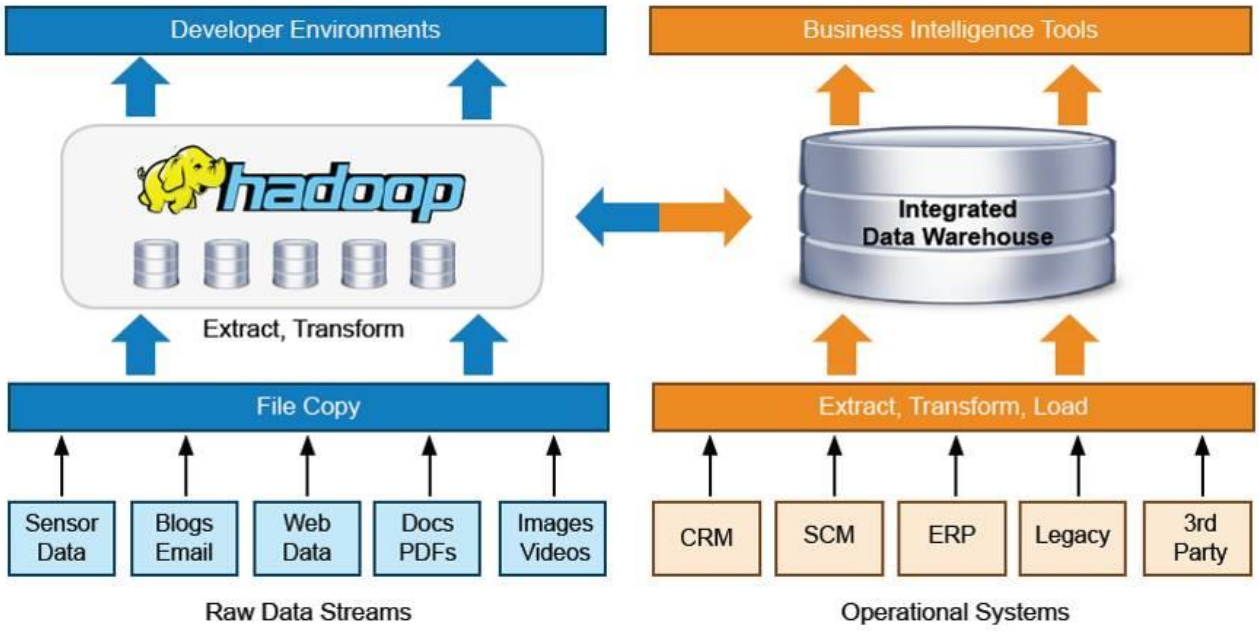
- SQL on Hadoop 是必须的。客户希望保持 SQL 接口的连续性。
- 混合数据仓库架构：针对不同的业务采用不同存储方案（Oracle 和 HDFS），数据量大的采用 HDFS 存储，数据量不够大的（不存在扩展性挑战的）可以依然使用关系型数据库。
- 逆规范化对性能的影响重大。通过对逆规范设计，可以达到关系数据库的查询性能。但是对于逆规范化是否存在其他影响，还需要研究。
- 相对于 sequence files 和 RC files，ORC 文件格式的性能是最好的。
- 实时 pipe 使用 storm 和 Kafka 实现。

就像 NewSQL 那样，可以有 New Data Warehouse 的。就是 Data Warehouse 与云计算的融合，即数据仓库的存储层在云平台，采用分布式系统。对应用侧而言， 原有的方式依旧有效，这样就不会资产浪费，而是有效的继承， 也是通往数据湖的一个较稳妥的步骤。

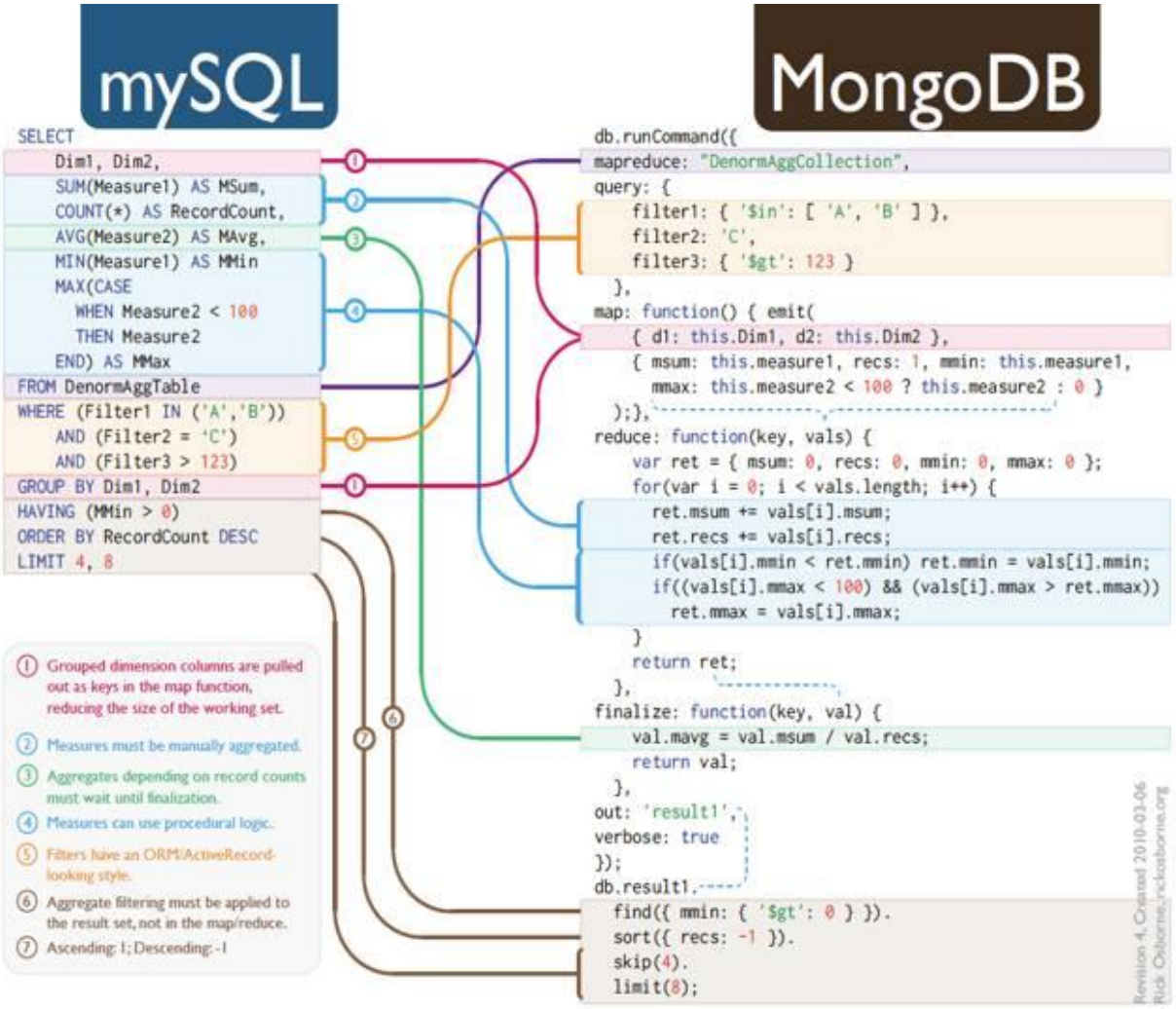
——曹洪伟

老曹这么一说，豁然开朗。我们在谈数据仓库架构向大数据架构演进的时候，其实我们在谈 New Data Warehouse 架构。就像当初数据仓库的出现是对数据库系统存在的限制进行补充一样，目前的大数据平台是对数据仓库系统存在的问题进行补充。他们的技术思路，技术架构，用户需求某种程度上是一致的，或者说核心的思想是一致的。不一致的地方仅仅是为了满足性能而做的技术方案的调整。

首先看数据集成架构。如下图，基于 Hadoop 的数据集成架构和基于关系数据库的传统数据集成架构是一致的。不同地方在于由于数据量的增大，左边的架构采用具有逆正则化（逆规范化）和顺序存储，不可更改数据集等特点的 Hadoop 平台存储数据。



其次看数据分析方法。虽然说基于 Hadoop 的数据集成架构采用了 Hadoop 数据存储平台（内置 MapRdecue 数据处理引擎），其数据操作，数据分析方法在思想上是一致的——从大量的数据集中获得由价值的信息——如下图所示，数据仓库的操作语句（group-by-aggregation）与 MapRdecue 的操作函数对应关系。



所以 MapRdecue 的核心思想就是在数据分片的基础上把数据仓库中的 group-by-aggregation 操作转换成分布式执行，MapRdecue 和传统数据仓库的思想是一致的。

- *The Map-Reduce programming model provides a good abstraction of group-by-aggregation operations over a cluster of machines.*
- *The programmer provides a map function that performs grouping and a reduce function that performs aggregation.*
- *The underlying run-time system achieves parallelism by partitioning the data and processing different partitions concurrently using multiple machines.*

所谓创新，继承和发展，大概如此吧。怪不得 Michael Stonebraker 撰文《MapReduce: A major step backwards》指出 MapReduce 是一个巨大倒退，并引发了他和 DeWitt 之间的大论战。Google 在 2010 年还为 MapRdecue 申请了专利，但我认为 MapReduce 不算是重大基础性创新，本质上还是云时代的数据仓库技术（New Data Warehouse）。但其作为 Google 三架马车的风头让人们大大忽略了

传统数据仓库的技术思想，误导了很多年轻学子的技术崇拜。所以本文尝试提供一个技术脉络：Data Warehouse->New Data Warehouse->Data Lake，阐述大数据技术背后的技术架构演进，抛砖引玉，欢迎批评指正。

Michael Stonebraker 对 MapReduce 的批评：

- *A giant step backward in the programming paradigm for large-scale data intensive applications.*
- *Not novel at all -- it represents a specific implementation of well known techniques developed nearly 25 years ago.*
- *To draw an analogy to SQL, map is like the group-by clause of an aggregate query. Reduce is analogous to the aggregate function (e.g., average) that is computed over all the rows with the same group-by attribute.*

在 New Data Warehouse 架构的基础上，如何向 Data Lake 演进？对电信行业来说，NFV 和 SDN 正在推动电信网络设备控制平面和数据平面的分离，电信设备数据会走向数据湖架构。电信设备数据融合，运营数据融合，最终会走向一个大融合。总结起来，电信大数据对于数据湖架构的拥抱，来自于以下四个方面的驱动。我用四个推导公式，如下：

- 5G->BigData (Semi-Structured and Unstructured) -> Modern Data Architecture for Enterprise -> Data Lake Storage Architecture -> Data Lake
- Cloud -> Network Function Cloudification -> Network Function Virtualization -> stateless VNF -> Distributed Sharing Storage -> Data Lake
- Distributed analytics -> Data Lake
- Hierarchy architecture -> Flat operations architecture -> Data Lake

我们尝试过在数据加载过程中自学习的产生数据库 schema，证明这个思路是可行的。基于结构化的数据，这个过程非常容易。但对于非结构化的数据，还是存在很大的挑战。使用机器学习的方式，模型训练成本恐怕和人工抽取 schema 的工作量是相当大的。但是我也看到在一些 CMDB 的数据库宣称已经支持数据库 schema 的自动升级，需要进一步调研。