Agenda 大綱

- 企業巨量資料生態架構規劃心法
- 建立生產歷程的常見痛點
  - 資料來源雜
  - 資料表太寬
  - 查詢速度慢
- 導入相關技術的前置行動
  - 目前關聯式資料庫的極限是幾個欄位?
  - 未來如何驗證評測解決方案? 如何產生範例資料集 ?

一頁道盡企業導入巨量資料的規劃心法秘笈
Enterprise Data Lake in One Page

導入藍圖 Roadmap 4

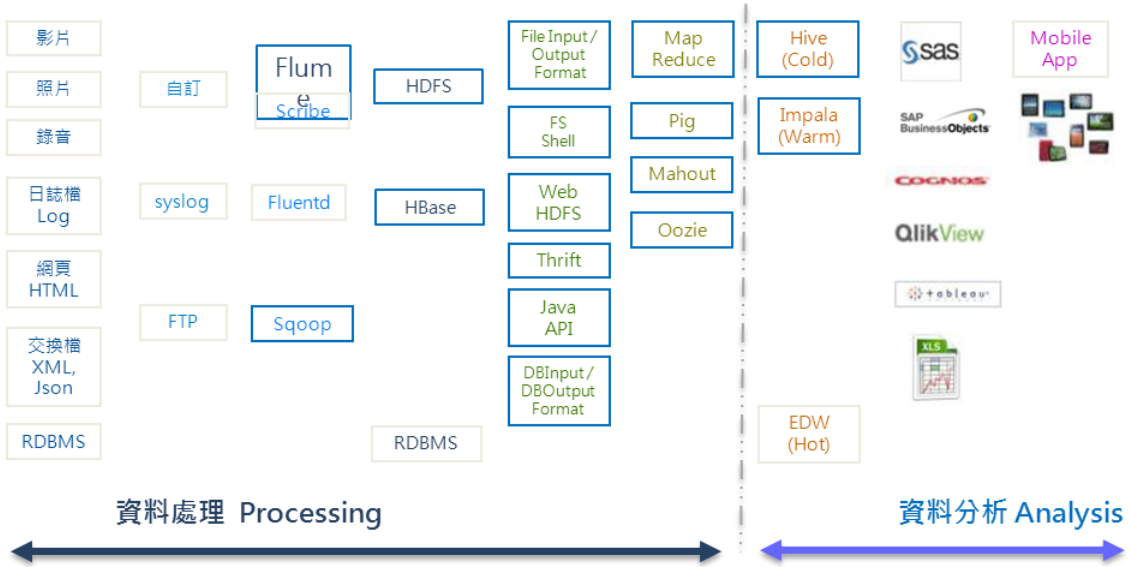
企業內部的人力資源盤點 People 1

處理巨量資料的常見流程 Process 2

生 流 蒐 存 取 算 析 用 看 變

資料源 網路協定 前處理 儲存方式 存取方式 資料處理 資料分析 視覺化 解讀 行動

處理巨量資料的技術盤點 Technology 3




Engineer
(電機)


Network
(網通)


System
Admin


Programmer
(資工)


DBA
(資管)


Analyst
(統計)


Decision
Maker

生 流 蒐 存 取 算 析 用 看 變

資料源 網路協定 前處理 儲存方式 存取方式 資料處理 資料分析 視覺化 解讀 行動

影片
照片
錄音
日誌檔 Log
網頁 HTML
交換檔 XML, Json
RDBMS

自訂
syslog
FTP

Flume
Scribe
Fluentd
Sqoop

HDFS
HBase
RDBMS

File Input / Output Format
FS Shell
Web HDFS
Thrift
Java API
DBInput / DBOutput Format

Map Reduce
Pig
Mahout
Oozie

Hive
Impala
EDW

Mobile App

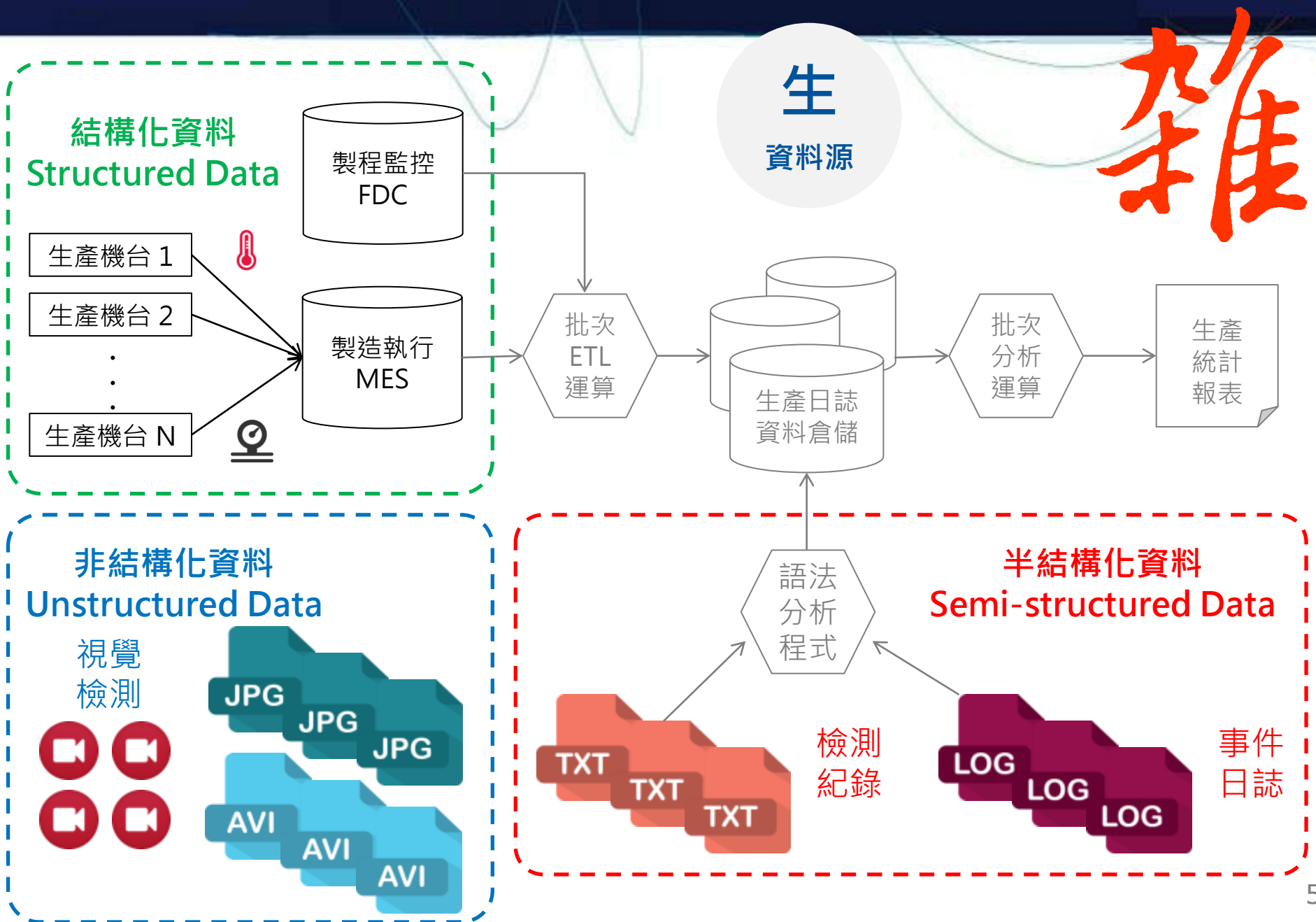

Etu Data Lake

資料處理 Processing

資料分析 Analysis

Agenda 大綱

- 企業巨量資料生態架構規劃心法
- 建立生產歷程的常見痛點
  - 資料來源雜
  - 資料表太寬
  - 查詢速度慢
- 導入相關技術的前置行動
  - 目前關聯式資料庫的極限是幾個欄位?
  - 未來如何驗證評測解決方案? 如何產生範例資料集 ?


生
資料源
雜
結構化資料
Structured Data
製程監控
FDC
生產機台 1
生產機台 2
生產機台 N
製造執行
MES
批次
ETL
運算
生產日誌
資料倉儲
批次
分析
運算
生產
統計
報表
非結構化資料
Unstructured Data
視覺
檢測
JPG
JPG
JPG
AVI
AVI
AVI
語法
分析
程式
半結構化資料
Semi-structured Data
TXT
TXT
TXT
檢測
紀錄
LOG
LOG
LOG
事件
日誌

固定格式的 Log – 耗時等級: 中等

- 通常以純文字格式存在,通常有固定的分隔符號跟欄位定義
- **常見痛點**：檔案太大，單機跑太久
- **目前解法**：平行切割檔案，平行解析內容
- 常見來源：
  - Windows Event Log (串流)
  - 能夠提供標準 Syslog 格式的網路設備 (串流)
  - XML 交換資料檔 (靜態)
  - 網頁伺服器 Web Access Log / DNS Access Log
- **常用工具**： **Pig** , Perl, Python
- **可批次化**： 易

魔鬼細節

流
網路協定

蒐
前處理

```
59.126.110.102 - - [01/Mar/2015:00:00:00 +0800] "GET
/action?;act=view;uid=;pid=0004880654;cat=J,J_007,J_007_009,J_007_009_016;erUid=f05a803a-b641-ebb0-
1fe1-383669dd375; HTTP/1.1" 302 160 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.7; rv:23.0)
Gecko/20100101 Firefox/23.0"
61.228.63.117 - - [01/Mar/2015:00:00:00 +0800] "GET
/action?;act=view;uid=;pid=0014166460;cat=L,L_010,L_010_006,L_010_006_016;erUid=c3931c39-87c5-78d4-
c7bc-33ef22963491; HTTP/1.1" 302 160 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/29.0.1547.76 Safari/537.36"
```

自由格式的 TXT – 耗時等級: 高等

- 通常以純文字格式存在,只有區塊定義
- **常見痛點**：檔案太多，單機跑太久
- **目前解法**：平行分配解析運算到多台
- 常見來源：
  - 晶片封裝檢測程式產生的日誌檔
  - 晶片模擬軟體產生的日誌檔
- **常用工具**： Perl, Python + **Hadoop Streaming**
- **可批次化**： 難，須注意例外處理

魔鬼細節

流
網路協定

蒐
前處理

```
LOTID:    XXXXXX        PID:  YYYYYY     MID: ZZZZZZZ
DATE:     2015/05/20    TIME:15:47:06    OP:  Jazz Wang     PROGRAM:     TEST-2

FUNC 1                                   FUNC 2
-------------------------                -------------------------
PIN1,V1,V2,V3,V4                         PIN4,V1,V2
PIN2,V1,V2,V3,V4,V5,V6,V7,V8,V9          PIN8,V3,V4,V5,V6,V7,V8
PIN3,V1,V2                               PIN3,V3,V5,V8,V7,V6
-------------------------                PIN2,V1,V3,V5
                                         PIN3,V1,V2,V3,V4
```

寬

存

儲存方式

結構化資料
Structured Data

製程監控
FDC

生產機台 1

生產機台 2

生產機台 N

製造執行
MES

批次
ETL
運算

生產日誌
資料倉儲

批次
分析
運算

生產
統計
報表

非結構化資料
Unstructured Data

視覺
檢測

JPG JPG JPG

AVI AVI AVI

半結構化資料
Semi-structured Data

語法
分析
程式

TXT TXT TXT

檢測
紀錄

LOG LOG LOG

事件
日誌

巨量資料時代，企業資料倉儲面臨的挑戰：寬

- 為了能夠做更多分析，需要建立更多連結資料 (Linked Data)
- 常見解法： JOIN 成一張更大的表
- **常見痛點**： 欄位**太寬**，資料量**太大**，寫入**太久**，單機跑不動
  叢集式資料倉儲又**太貴**
- 問題根源：
  - **太寬** → 現有資料倉儲是以關聯式資料庫實作,本質上會建議拆成多張表
  - **太大** → 關聯式資料庫必須靠 Sharding 來解決資料量的問題
  - **太久** → 關聯式資料庫是**為了查詢快**，所以寫入時要**花 CPU 建立索引**
- **目前解法**：
  - **入庫即查** (Schema on Read 解決太寬問題, 不建索引自然寫入快)
  - **平行寫入** (分散儲存解決資料量問題, 平行寫入解決寫入太久問題)
- **常用工具**： **Hive** (SQL-like), **HBase** (NoSQL)
- **未來挑戰**： 遞增資料的入庫

寬跟慢是一體兩面：能有多寬? 能有多大? 會有多慢?

- 快或慢，很主觀。
- 回歸商業問題的需求，再來定義對速度的要求
  - 批次 (年/月/週/日) – 近即時 (時/分) – 即時 (秒/毫秒)
- 資料集特性：
  - 總欄位數 = 製程工序數 (column) or 零售商品數
  - 總資料量 = 製程工序數 * 資料型態大小 * 生產數量(row)
- 請發揮您的想像力： (因為這些我也不知道答案,只能讓各位感受與想像一下)
  - 如果今天有 3,000 個連線要**同時**將資料寫入關聯式資料庫
  - SQL Client 同時連線數的上限 ?? 需要購買多少**授權** ??
  - 需要買幾核心的機器呢 ?? 多少記憶體 ?? 該用光纖接 SAN 嗎 ??
  - **硬碟寫入速度**該有多快 ?? 會不會發生競爭同一顆硬碟的問題 ??

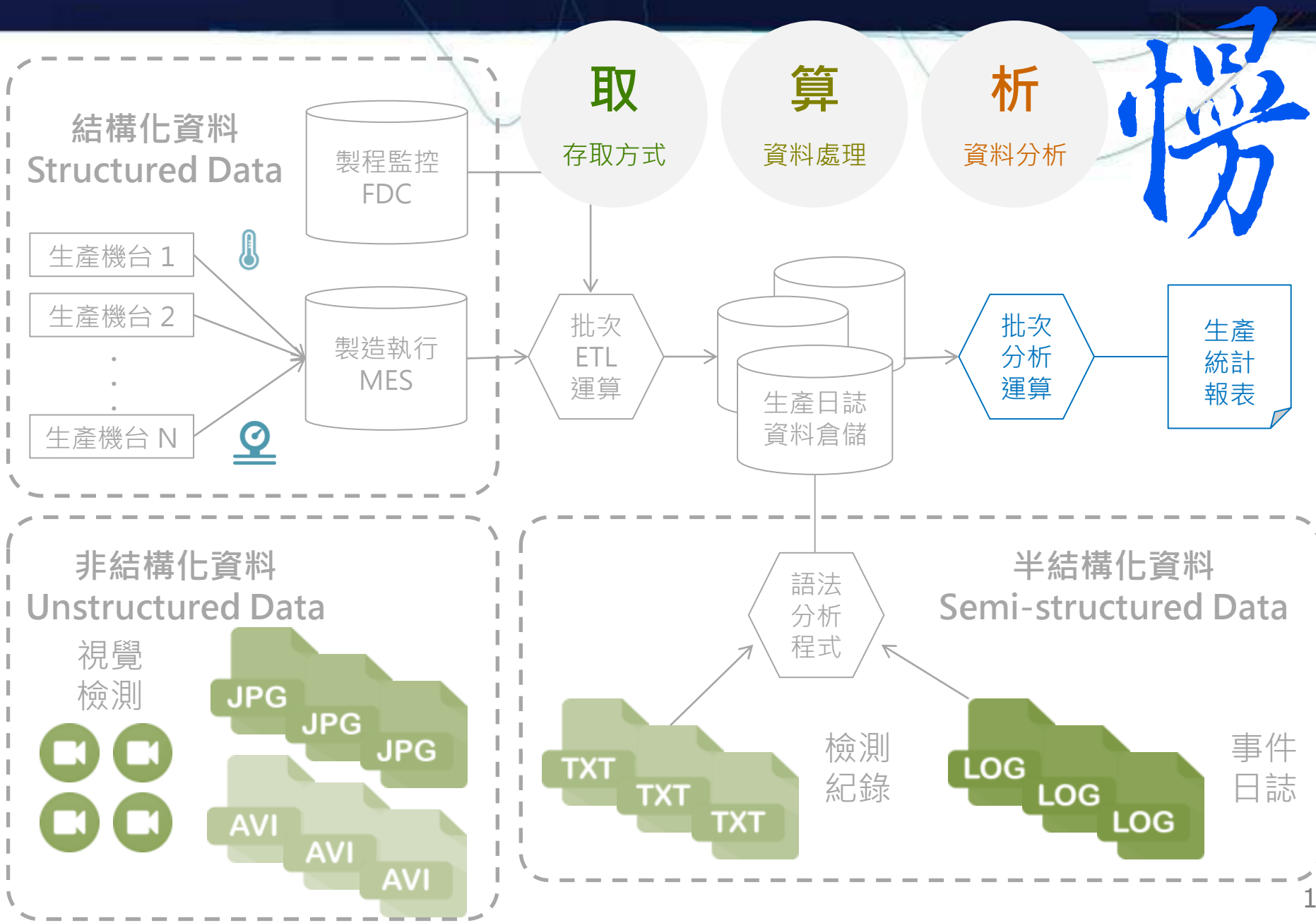
避免落入效能規格戰爭
產生商業價值才是重點

示意：客戶特徵分析 (原始)

	商品1	商品2	商品3	商品4	商品5	商品6	商品7	商品8	商品9
客戶A	金額1				金額1				
客戶B		金額2			金額1				
客戶C			金額3			金額1			
客戶D	金額2			金額4					
客戶E		金額3		金額5	金額5				
客戶F			金額4			金額6			
客戶G			金額5				金額7		
客戶H		金額6		金額7				金額8	
客戶I			金額8						金額9
客戶J		金額9		金額1					
客戶K			金額9		金額2				
客戶L						金額3			

示意：客戶特徵分析 (分群)

	商品1	商品2	商品3	商品4	商品5	商品6	商品7	商品8	商品9
客戶A	金額1				金額1				
客戶D	金額2			金額4					
客戶B		金額2			金額1				
客戶E		金額3		金額5	金額5				
客戶H		金額6		金額7				金額8	
客戶J		金額9		金額1					
客戶C			金額3			金額1			
客戶F			金額4			金額6			
客戶G			金額5				金額7		
客戶I			金額8						金額9
客戶K			金額9		金額2				
客戶L						金額3			


取
存取方式
算
資料處理
析
資料分析
結構化資料
Structured Data
製程監控
FDC
生產機台 1
生產機台 2
生產機台 N
製造執行
MES
批次
ETL
運算
生產日誌
資料倉儲
批次
分析
運算
生產
統計
報表
非結構化資料
Unstructured Data
視覺
檢測
JPG
JPG
JPG
AVI
AVI
AVI
語法
分析
程式
半結構化資料
Semi-structured Data
TXT
TXT
TXT
檢測
紀錄
LOG
LOG
LOG
事件
日誌

常見痛點：查詢太慢


DBA
(資管)


Analyst
(統計)

- **人與技術：**
  - 通常組織內部是由熟悉 SQL 與統計語言的資料庫管理員與分析師在進行分析
  - 為了解釋現象與原因，會利用視覺化工具，繪製「管制圖」方便解讀，產生洞察
- **常見痛點：**
  - **複雜 SQL 查詢**執行**太久**
  - 報表反應速度**很慢**
- **一般解法：**
  - **查詢太久** → (下一頁)
  - **反應很慢** → 預處理 (先建好要查的資料表)

```
select
  i_item_id,  s_state, avg(ss_quantity) agg1,
  avg(ss_list_price) agg2, avg(ss_coupon_amt) agg3,
  avg(ss_sales_price) agg4
from
  store_sales,  customer_demographics,
  date_dim,  store,  item
Where
  ss_sold_date_sk = d_date_sk
  and ss_item_sk = i_item_sk
  and ss_store_sk = s_store_sk
  and ss_cdemo_sk = cd_demo_sk
  and cd_gender = 'F'
  and cd_marital_status = 'W'
  and cd_education_status = 'Primary'
  and d_year = 1998
  and s_state in ('WI', 'CA', 'TX', 'FL', 'WA', 'TN')
  and ss_sold_date_sk between 2450815 and 2451179
group by
  i_item_id, s_state
order by
  i_item_id, s_state
limit 100;
```

取
存取方式

算
資料處理

痛點：查詢太慢，那怎麼找止痛藥呢？

- **進階解法：**
  - 查詢平行執行（ SQL Execution Plan 平行化分配到不同節點運算 ）
  - 用記憶體加速（ In Memory Database ）
  - 資料結構最佳化（ 善用 Partition, 改用 Parquet 格式, 線性壓縮 ）
  - 查詢語句最佳化（ 觀察 SQL Execution Plan , 調整系統參數 ）
- **常用工具：**
  - **互動式查詢：Impala**（ In-Memory Database, SQL-92 Ad-Hot Query ）
  - **大範圍查表：HBase**（ In-Memory Datastore , 只有 PUT/GET/SCAN)
  - 大範圍分析：**Hive**（ Batch, 數分到數天, SQL-92 ）
- **標準介面： ODBC / JDBC**
- **潛在挑戰：** SQL 語句的移轉

取
存取方式

算
資料處理

痛點：分析太慢，那止痛藥呢？

析

資料分析

- **新興需求：多變量分析、機器學習**
  - 分析方法通常會使用到機器學習的工具，
    如：降維(PCA/SVD)、分群(K-Mean)等
  - 若考量人與技術：統計軟體 R 與 SQL 仍是主流
- **常見痛點：**
  - **單機記憶體不足、運算時間太久、讀取資料太慢**
- **目前解法：**
  - 不足 → 平行運算（拆解資料，將運算平行化分配到不同節點）
  - 太久 → 善用記憶體加速（In Memory Computing）
  - 太慢 → 平行讀寫（有迭代特性的演算法, 盡量減少中間產物）
- **常用工具：**
  - 批次機器學習：　**Mahout**　（Batch）
  - 機器學習函式庫：　**Spark MLlib**　（In-Memory Computing , RDD）

Agenda 大綱

- 企業巨量資料生態架構規劃心法
- 建立生產歷程的常見痛點
  - 資料來源雜
  - 資料表太寬
  - 查詢速度慢
- 導入相關技術的前置行動
  - 目前關聯式資料庫的極限是幾個欄位?
  - 未來如何驗證評測解決方案? 如何產生範例資料集 ?

企業無法外包的部分：用、看、變

- 資訊服務業通常熟悉技術，也有具備對應的技術人才

  然而很難取代企業內部的分析專才，

  因為『**欠缺領域知識(Domain Knowledge)**』。
- 企業常常只有老師傅才有辦法『判讀』，

  能否把『經驗』轉化成『系統』呢？
- **前期規劃 vs 付諸行動**

用
視覺化

看
解讀

變
行動

價值 行動

目前關聯式資料庫的極限是幾個欄位?

- 確認了商務需求與產出價值後，下一步是確認是否需要擁抱巨量資料
- 以下是目前常用的關聯式資料庫欄位寬度限制：

Database	Maximum Columns Per table
Oracle	1,000
SQL Server	1,024 columns per nonwide table
	30,000 columns per wide table
MySQL	4,096
Java Derby	1,012

[1] https://docs.oracle.com/cd/B28359_01/server.111/b28320/limits003.htm#i288032
[2] https://msdn.microsoft.com/en-us/ms143432.aspx
[3] https://dev.mysql.com/doc/refman/5.6/en/column-count-limit.html
[4] http://docs.oracle.com/javadb/10.8.3.0/ref/rrefdbmlimits.html

實測 MySQL – 建立單表 4,000 個整數欄位 <失敗>

- 資料型態、是否允許空值(NULL)、引擎種類均會影響欄位個數

```
root@mysql:/vagrant# mysql -u root -p < column_limit_4K.sql
Enter password:
ERROR 1117 (HY000) at line 3: Too many columns
root@mysql:/vagrant# cat column_limit_4K.sql
CREATE DATABASE IF NOT EXISTS test;
USE test;
CREATE TABLE IF NOT EXISTS 4K_columns ( col1 int, col2 int, col3 int, col4 int, col5 int, col
 int, col12 int, col13 int, col14 int, col15 int, col16 int, col17 int, col18 int, col19 int,
t, col25 int, col26 int, col27 int, col28 int, col29 int, col30 int, col31 int, col32 int, co
col38 int, col39 int, col40 int, col41 int, col42 int, col43 int, col44 int, col45 int, col46
51 int, col52 int, col53 int, col54 int, col55 int, col56 int, col57 int, col58 int, col59 in
int, col65 int, col66 int, col67 int, col68 int, col69 int, col70 int, col71 int, col72 int,
, col78 int, col79 int, col80 int, col81 int, col82 int, col83 int, col84 int, col85 int, col
ol91 int, col92 int, col93 int, col94 int, col95 int, col96 int, col97 int, col98 int, col99
col104 int, col105 int, col106 int, col107 int, col108 int, col109 int, col110 int, col111 in
l116 int, col117 int, col118 int, col119 int, col120 int, col121 int, col122 int, col123 int,
28 int, col129 int, col130 int, col131 int, col132 int, col133 int, col134 int, col135 int, c
 int, col141 int, col142 int, col143 int, col144 int, col145 int, col146 int, col147 int, col
nt, col153 int, col154 int, col155 int, col156 int, col157 int, col158 int, col159 int, col16
, col165 int, col166 int, col167 int, col168 int, col169 int, col170 int, col171 int, col172
```

[1] column_limit_4K.sql - https://gist.github.com/601d039c8b754c661220
[2] column_limit_5K.sql - https://gist.github.com/c9e71a2a051ccd4ac418

實測 Impala – 建立 5,000 個整數欄位 <成功>

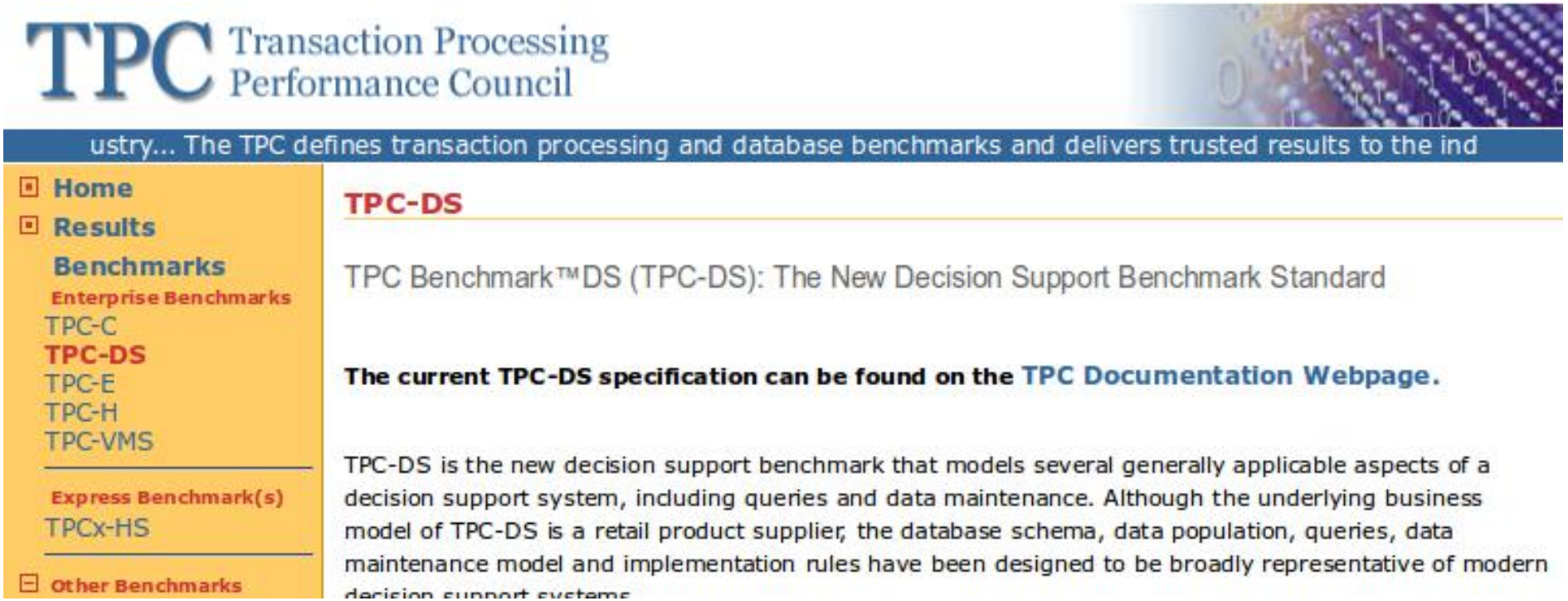
```
[root@master ~]# impala-shell -f column_limit_5K.sql
Starting Impala Shell without Kerberos authentication
Connected to master:21000
Server version: impalad version 2.1.0-cdh4 RELEASE (build 11a45f84eb1f0d441ebad72cf9d65262f6cc2391)
Query: create DATABASE IF NOT EXISTS test

Fetched 0 row(s) in 1.05s
Query: use test
Query: create TABLE IF NOT EXISTS 10K_columns ( col1 int, col2 int, col3 int, col4 int, col5 int, col6 int, col7 i
t, col11 int, col12 int, col13 int, col14 int, col15 int, col16 int, col17 int, col18 int, col19 int, col20 int, c
col24 int, col25 int, col26 int, col27 int, col28 int, col29 int, col30 int, col31 int, col32 int, col33 int, col3
37 int, col38 int, col39 int, col40 int, col41 int, col42 int, col43 int, col44 int, col45 int, col46 int, col47 i
int, col51 int, col52 int, col53 int, col54 int, col55 int, col56 int, col57 int, col58 int, col59 int, col60 int,
, col64 int, col65 int, col66 int, col67 int, col68 int, col69 int, col70 int, col71 int, col72 int, col73 int, co
ol77 int, col7                                                t, col82 int, col83 int, col84 int, col85 int, col86 int, col87
0 int, col91 i                                                col95 int, col96 int, col97 int, col98 int, col99 int, col100 i
03 int, col104                                                int, col108 int, col109 int, col110 int, col111 int, col112 int
 int, col116 i                                                t, col120 int, col121 int, col122 int, col123 int, col124 int,
 col4923 int, col4924 int, col4925
l4934 int, col4935 int, col4936 in
45 int, col4946 int, col4947 int,
int, col4957 int, col4958 int, col
, col4968 int, col4969 int, col497
ol4979 int, col4980 int, col4981 i
990 int, col4991 int, col4992 int,

Fetched 0 row(s) in 4.89s
```

[1] column_limit_4K.sql - https://gist.github.com/601d039c8b754c661220
[2] column_limit_5K.sql - https://gist.github.com/c9e71a2a051ccd4ac418

未來如何驗證評測解決方案

- 目前坊間有非常多解決方案，橘子、蘋果該怎麼比較呢？
- 建議使用國際標準：TPC-DS

  http://www.tpc.org/tpcds/



TPC-DS 定義了評測用的資料表 Schema

```
[master:21000] > show databases;
Query: show databases
+------------------+
| name             |
+------------------+
| _impala_builtins |
| default          |
| tpcds            |
| tpcds_parquet    |
| tpcds_rcfile     |
+------------------+
Fetched 5 row(s) in 0.03s
[master:21000] > use tpcds;
Query: use tpcds
```

```
[master:21000] > show tables;
Query: show tables
+------------------------+
| name                   |
+------------------------+
| customer               |
| customer_address       |
| customer_demographics  |
| date_dim               |
| household_demographics |
| inventory              |
| item                   |
| promotion              |
| store                  |
| store_sales            |
| time_dim               |
+------------------------+
Fetched 11 row(s) in 0.01s
```

TPC-DS 也定義了評測用的 SQL 查詢

```
-- start query 1 in stream 0 using template query27.tpl
select
  i_item_id,
  s_state,
  -- grouping(s_state) g_state,
  avg(ss_quantity) agg1,
  avg(ss_list_price) agg2,
  avg(ss_coupon_amt) agg3,
  avg(ss_sales_price) agg4
from
  store_sales,
  customer_demographics,
  date_dim,
  store,
  item
Where
  ss_sold_date_sk = d_date_sk
  and ss_item_sk = i_item_sk
  and ss_store_sk = s_store_sk
  and ss_cdemo_sk = cd_demo_sk
  and cd_gender = 'F'
  and cd_marital_status = 'W'
  and cd_education_status = 'Primary'
  and d_year = 1998
  and s_state in
      ('WI', 'CA', 'TX', 'FL', 'WA', 'TN')
  and ss_sold_date_sk between 2450815 and 2451179
  -- partition key filter
group by
  -- rollup (i_item_id, s_state)
  i_item_id,
  s_state
order by
  i_item_id,
  s_state
limit 100;
-- end query 1 in stream 0 using template query27.
```

TPC-DS 也有提供產生指定筆數資料的工具

取樣資料表：資料量 3.8G, 文字檔格式, 沒有壓縮, 3 千萬筆資料

```
[master:21000] > show table stats customer;
Query: show table stats customer
+-------+--------+--------+--------------+--------+-------------------+
| #Rows | #Files | Size   | Bytes Cached | Format | Incremental stats |
+-------+--------+--------+--------------+--------+-------------------+
| -1    | 1      | 3.81GB | NOT CACHED   | TEXT   | false             |
+-------+--------+--------+--------------+--------+-------------------+
Fetched 1 row(s) in 0.00s
[master:21000] > select count(*) from customer;
Query: select count(*) from customer
+----------+
| count(*) |
+----------+
| 30000000 |
+----------+
Fetched 1 row(s) in 0.77s
```

使用相同的資料集與
查詢語句，較容易進行
不同技術的評選

Take Away – 今天的分享總結

1. 規劃心法：People、Process、Technology、Roadmap

生	流	蒐	存	取	算	析	用	看	變
資料源	網路協定	前處理	儲存方式	存取方式	資料處理	資料分析	視覺化	解讀	行動

2. RDBMS 與 NoSQL/NewSQL/SQL on Hadoop 各有優勢
   想清楚為何需要改用新技術，切忌盲目追趕新技術。
   - 您是否遇到太寬、太雜、太慢的問題呢？
   - NoSQL/NewSQL/SQL on Hadoop 目前的發展是
     把 RDBMS 的組成元件拆解成不同專案

     建議聽一下 Rich Hickey 在 HadoopWorld 2012 的 Keynote
     https://youtu.be/P-NZei5ANaQ