高速屋
KOUSOKUYA


# 動作検証レポート

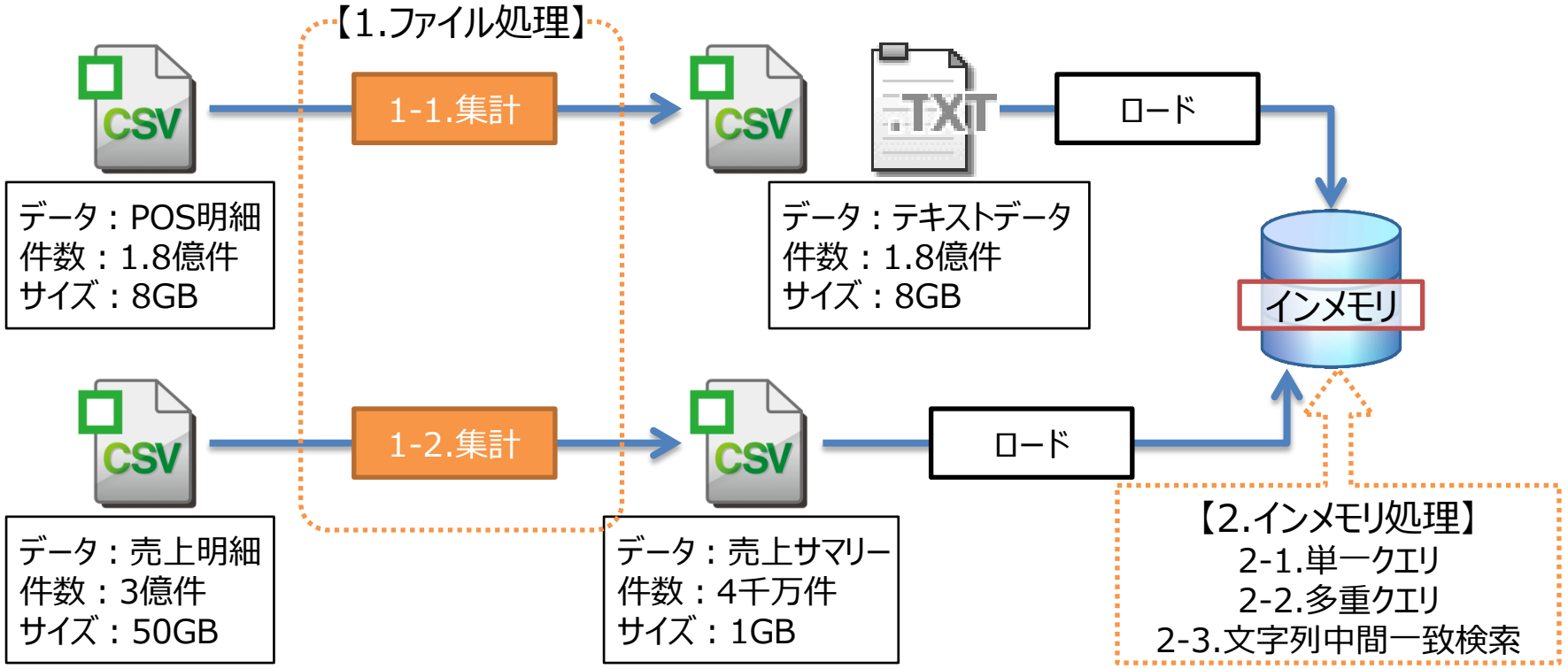
2016年6月

株式会社 高速屋

# 動作検証概要

**検証対象：**

【1.ファイル処理】　入力CSVファイルを読込み、処理結果をCSVファイルに出力
【2.インメモリ処理】　事前にインメモリ化されたデータに対してクエリ(Select文)を実行




検証対象において、 同一サーバー上で解析ブースターと市販DBMS(Microsoft SQL Server)の処理速度比較(ベンチマーク)を実施

# システム環境

使用機種

| | |
|---|---|
| 機種名 | NEC Express5800 / R110h-1 |
| CPU | Xeon 4C E3-1270v5 ×1 |
| クロック | 3.6GHz |
| コア数 | 4 |
| 論理プロセッサ数 | 8 |
| メモリ | 32GB |
| ハードディスク容量 | 1200GB（600GB × 3 Array） |
| OS | Windows Server 2012 R2 Std. |

ハードディスク性能（MB/s）

| アクセス | Read | Write |
|---|---|---|
| シーケンシャル 32Q | 452.846 | 412.162 |
| 4KB ランダム 32Q | 8.728 | 2.428 |
| シーケンシャル | 588.286 | 76.332 |
| 4KB ランダム | 1.32 | 0.559 |

（※CrystalDiskMark 4.0.3 x64 使用）

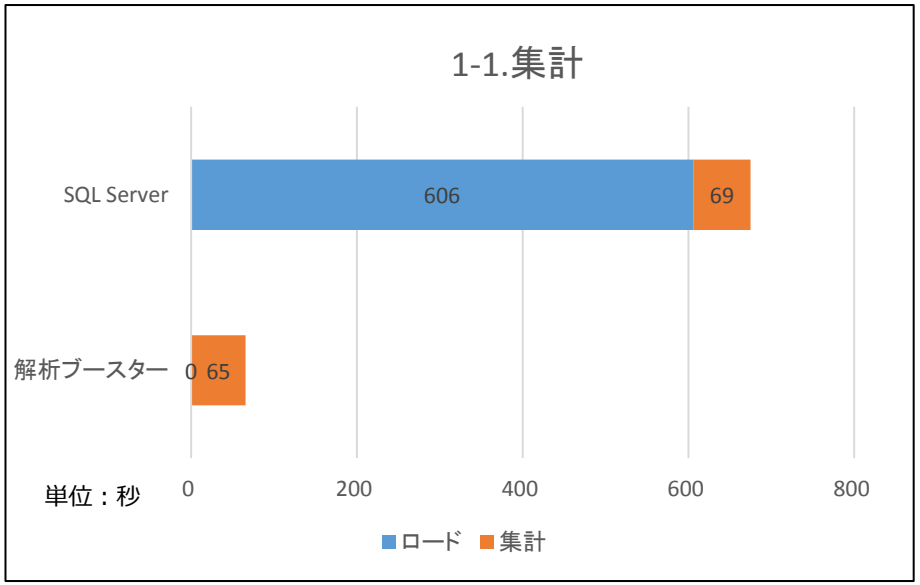
バージョン

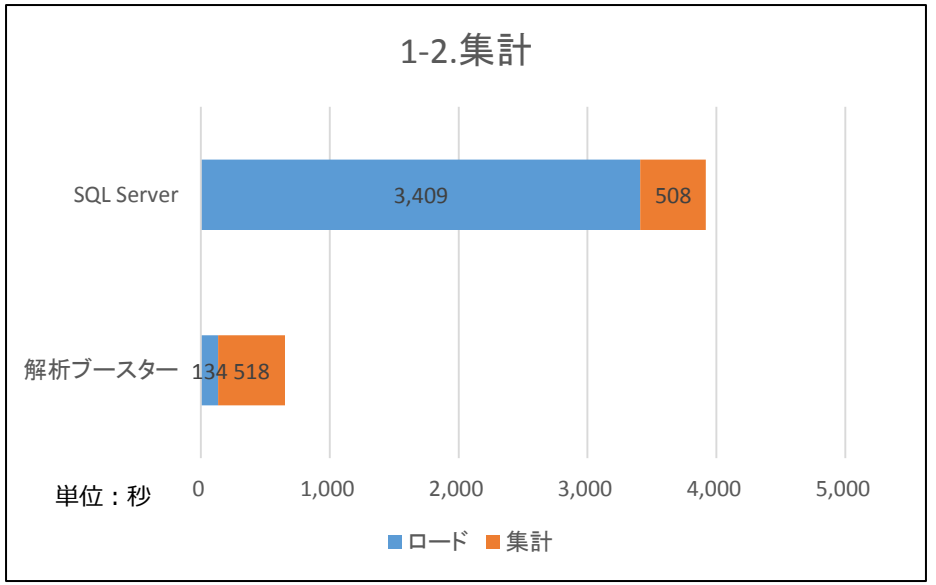
解析ブースター: BOOSTER_v2.0_6.5.0.60_UTF8_release6_20160430

SQL Server: Microsoft SQL Server 2014 - 12.0.2000.8 (X64)
Feb 20 2014 20:04:26
Copyright (c) Microsoft Corporation
Developer Edition (64-bit) on Windows NT 6.3 <X64> (Build 9600: )

# ◆検証結果要約　1.ファイル処理

**1-1. 件数 1.8億件、サイズ 8 GB のPOS明細集計**
**1-2. 件数 3億件、サイズ 50GB の売上集計**

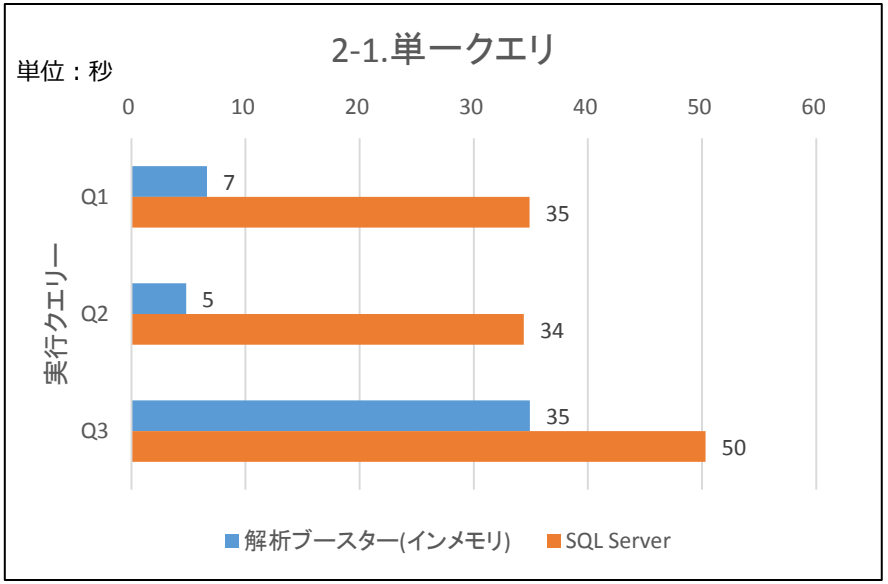


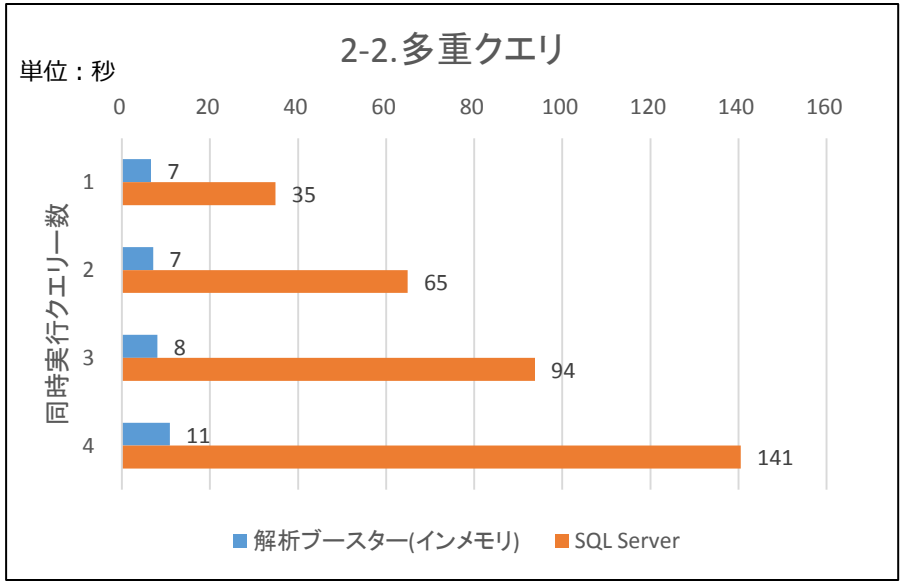




**解析ブースターは明細のロードをしなくてもSQL Serverと同等の集計時間！**

# ◆検証結果要約　2.インメモリ処理

**2-1. 件数 4千万件、サイズ 1 GB のインメモリデータに対する単一クエリ**
**2-2. 件数 4千万件、サイズ 1 GB のインメモリデータに対する多重クエリ**

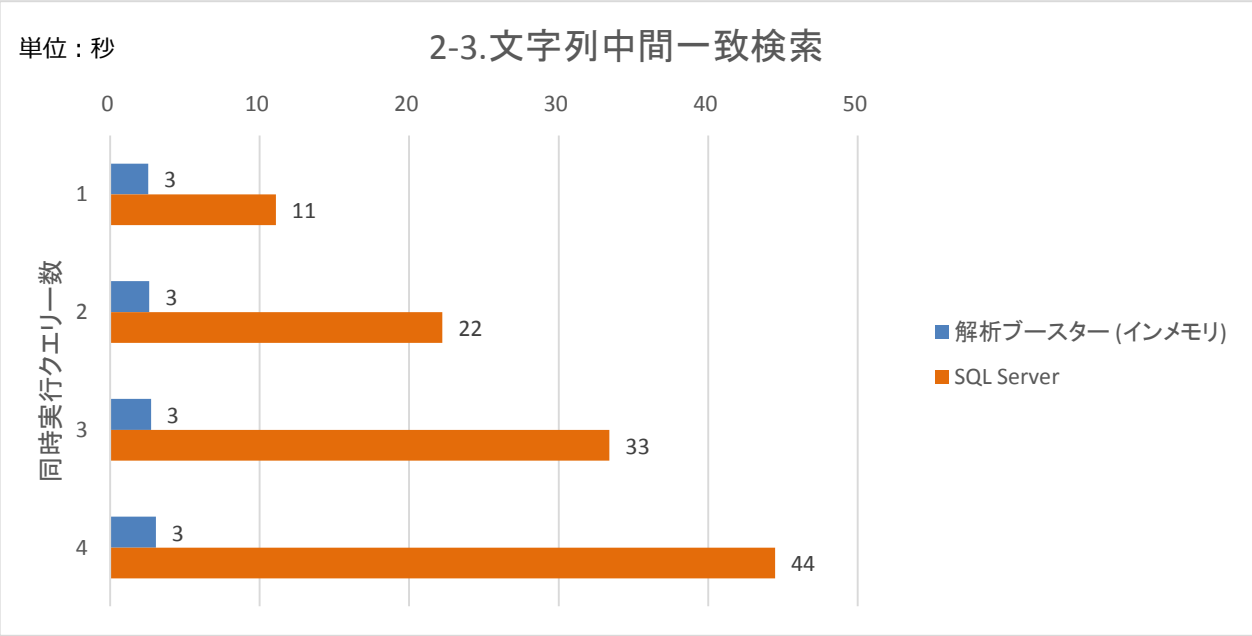



**解析ブースターのインメモリ処理はSQL Serverの2～5倍のハイレスポンスを実現 ！**
**解析ブースターのインメモリ処理は多重度を上げてもレスポンス劣化が軽微 ！**

# ◆検証結果要約　2.インメモリ処理

## 2-3. 件数 1億件、サイズ 20GBのテキストデータに対する文字列中間一致検索




**解析ブースターインメモリ処理はSQL Serverの5～15倍の検索性能を実現 ！**

高速屋
KOUSOKUYA

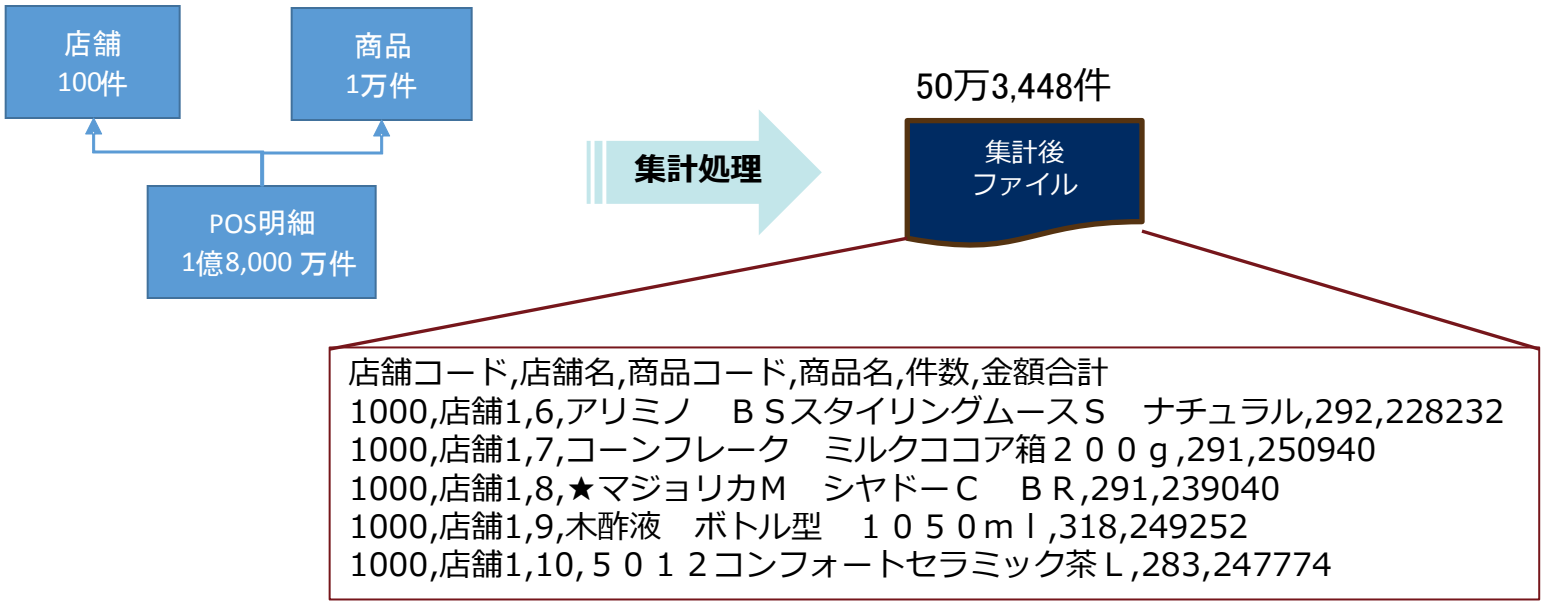

# 動作検証 ― 詳細編

# 1.ファイル処理

## 1-1. 件数 1.8億件、サイズ 8 GB のPOS明細集計

検証シナリオ： POSデータ 店舗別 商品別集計
下表のPOSデータを店舗別、商品別に件数と金額の集計処理を行う。
集計後の件数は50万3,448件となる。

| ファイル | 件数 | CSVサイズ(KB) |
|---|---|---|
| 店舗 | 100 | 7 |
| 商品 | 10,364 | 677 |
| POS明細 | 184,648,243 | 8,388,608 |



# 1.ファイル処理

## 1-1. 件数 1.8億件、サイズ 8 GB のPOS明細集計

実行結果　　単位：ミリ秒

| | ロード | 明細集計 | 店舗・商品結合 | 実行時間合計 |
|---|---|---|---|---|
| **解析ブースター** | 341 | 64,469 | 974 | 65,784 |
| **SQL Server** | 605,612 | 69,256 | | 674,868 |

処理手順の概要

ロード

- 解析ブースターのロードでは、店舗マスターと商品マスターを結合するために、明細データ以外のマスターデータを一時テーブルにロードします。
- SQL Serverのロードでは、明細データを含む全データをテーブルにロードします。

明細集計

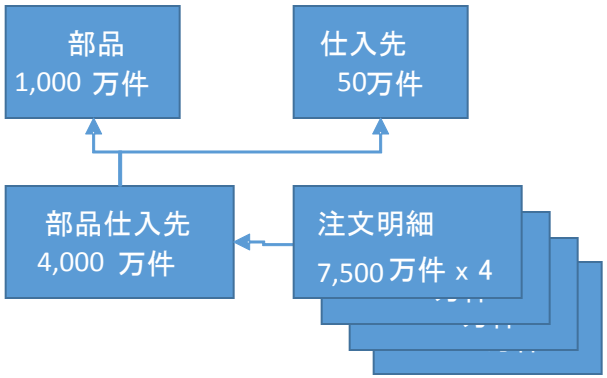
- 解析ブースターの明細集計では、先に明細データを単独集計して、その結果と店舗マスター、商品マスターを結合します。
- SQL Serverの明細集計では、店舗マスターと商品マスターの結合含め単一のSQL文で処理されます。

# 1.ファイル処理

## 1-2. 件数 3億件、サイズ 50GB の売上集計

検証シナリオ：部品別 仕入先別集計

下表のデータを部品別、仕入先別に件数と金額で集計処理を行う。
集計後の件数は3,998万4,471件となる。




| テーブル | 件数 | CSVサイズ(KB) |
|---|---|---|
| 部品 | 10,000,000 | 1,202,199 |
| 仕入先 | 500,000 | 70,187 |
| 部品仕入先 | 40,000,000 | 5,974,433 |
| 注文明細1 | 75,001,453 | 9,667,804 |
| 注文明細2 | 75,001,453 | 9,727,411 |
| 注文明細3 | 75,001,453 | 9,751,990 |
| 注文明細4 | 75,001,452 | 9,751,883 |

**集計処理**

3,998万4,471件

集計後
ファイル

```
部品名,仕入先名,商品コード,数量,金額
"goldenrod lavender spring chocolate lace","Supplier#000000002",9,221582.93
"goldenrod lavender spring chocolate lace","Supplier#000125002",4,49816.29
"goldenrod lavender spring chocolate lace","Supplier#000250002",8,211825.10
"goldenrod lavender spring chocolate lace","Supplier#000375002",6,176605.01
"blush thistle blue yellow saddle","Supplier#000000003",12,269968.60
```

# 1.ファイル処理

## 1-2. 件数 3億件、サイズ 50GB の売上集計

実行結果　　単位：ミリ秒

| | ロード | 明細集計 | 部品・仕入先結合 | 実行時間合計 |
|---|---|---|---|---|
| **解析ブースター** | 134,022 | 457,414 | 60,142 | 651,578 |
| SQL Server | 3,409,487 | 508,414 | | 3917,901 |

処理手順の概要

ロード

- 解析ブースターのロードでは、部品マスター及び仕入先マスターを結合するために、明細データ以外のマスターを一時テーブルにロードします。
- SQL Server のロードでは、明細データを含む全データをロードします。

明細集計

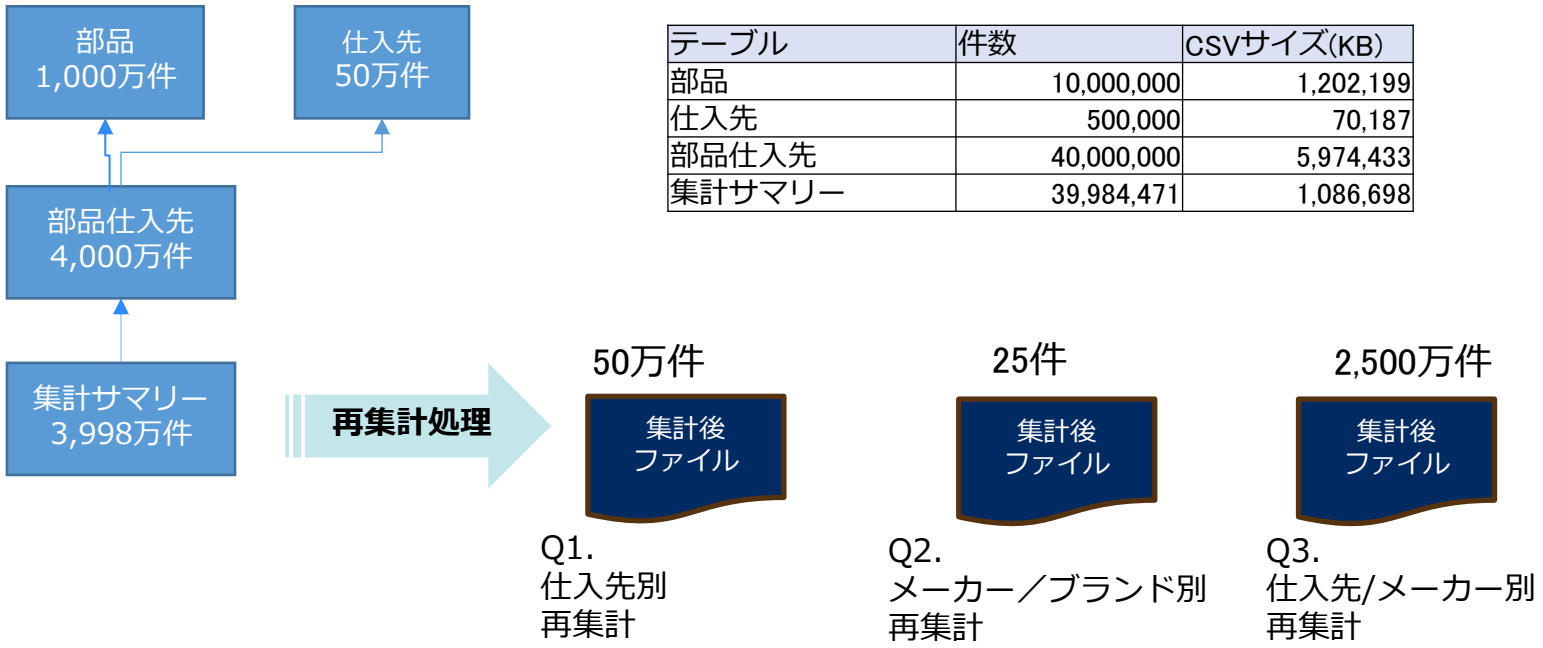
- 解析ブースターの明細集計では、先に明細データを単独集計して、その結果と部品マスター、仕入先マスターを結合します。
- SQL Server の明細集計では、部品マスター、仕入先マスターとの結合含め単一のSQL文で処理されます。

# 2.インメモリ処理

## 2-1.件数 4千万件、サイズ 1 GB のインメモリデータに対する単一クエリ

検証シナリオ：部品別、仕入先別に集計されたデータに対する再集計処理

あらかじめ部品別、仕入先別に件数と金額が集計されたデータに対して
下記、Q1,Q2,Q3の再集計処理を行う。




| テーブル | 件数 | CSVサイズ(KB) |
|---|---|---|
| 部品 | 10,000,000 | 1,202,199 |
| 仕入先 | 500,000 | 70,187 |
| 部品仕入先 | 40,000,000 | 5,974,433 |
| 集計サマリー | 39,984,471 | 1,086,698 |

# 2.インメモリ処理

## 2-1.件数 4千万件、サイズ 1 GB のインメモリデータに対する単一クエリ

実行結果　　単位：ミリ秒

| クエリーNo | 処理内容 | 解析ブースター（インメモリ） | SQL Server |
|---|---|---|---|
| Q1 | 仕入先別再集計 | 6,626 | 34,884 |
| Q2 | メーカー／ブランド別再集計 | 4,810 | 34,394 |
| Q3 | 仕入先/メーカー別再集計 | 34,920 | 50,314 |

処理手順の概要

ロード

- 解析ブースターもSQL Serverも、クエリー実行時にはロード済みのテーブルにのみアクセスします。

再集計

- 解析ブースターとSQL Serverは同等のクエリー(Select文)で再集計を実行します。

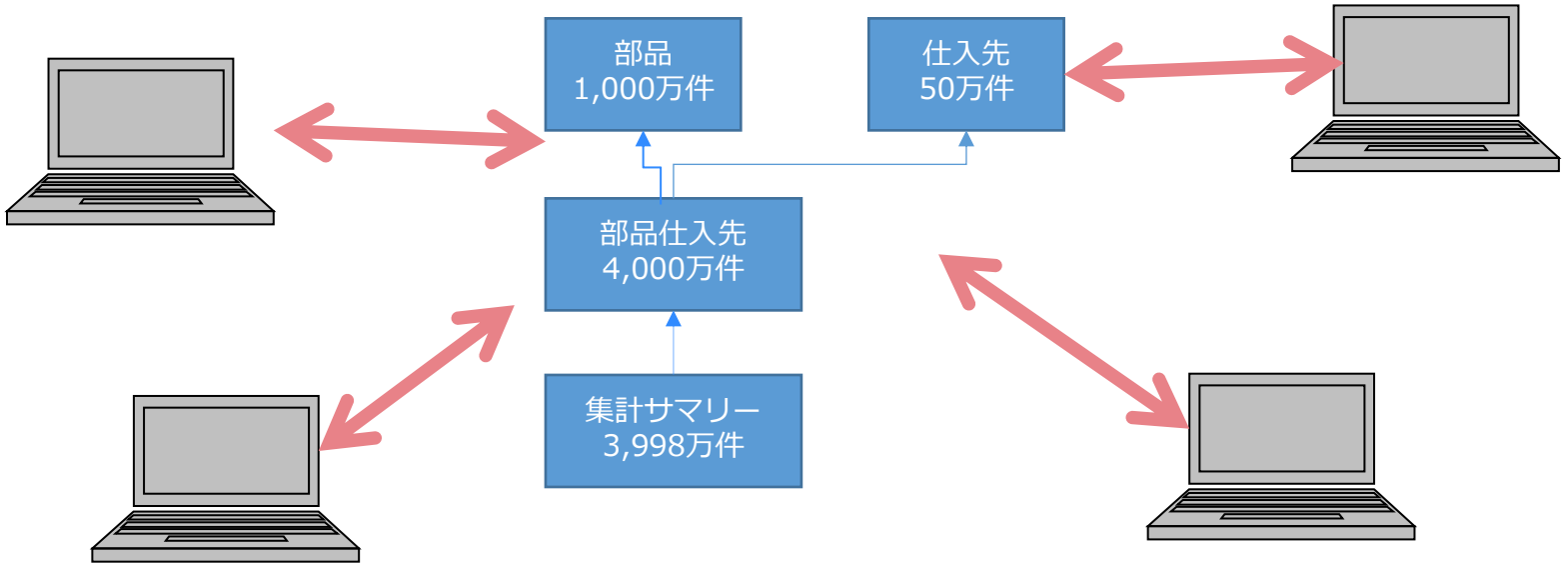
Q3のクエリ例

```
select
supplier.s_name as supp_name,part.p_mfgr as mfgr,
sum(lineitem_sum.total_cnt) as cnt,sum(lineitem_sum.total_sales) as sales
from lineitem_sum
inner join partsupp on lineitem_sum.partkey = partsupp.ps_partkey and
lineitem_sum.suppkey = partsupp.ps_suppkey
inner join part on partsupp.ps_partkey = part.p_partkey
inner join supplier on partsupp.ps_suppkey = supplier.s_suppkey
group by supplier.s_name, part.p_mfgr
```

# 2.インメモリ処理

## 2-2.件数 4千万件、サイズ 1 GB のインメモリデータに対する多重クエリ

検証シナリオ：クエリーQ1を同時実行数1～4で多重に実行する。



# 2.インメモリ処理

## 2-2.件数 4千万件、サイズ 1 GB のインメモリデータに対する多重クエリ

多重クエリー実行時間　　単位：ミリ秒

| 同時実行数 | 解析ブースター<br>(インメモリ) | SQL Server |
|---|---|---|
| 1 | 6,626 | 34,884 |
| 2 | 7,124 | 64,896 |
| 3 | 8,070 | 93,774 |
| 4 | 10,880 | 140,504 |

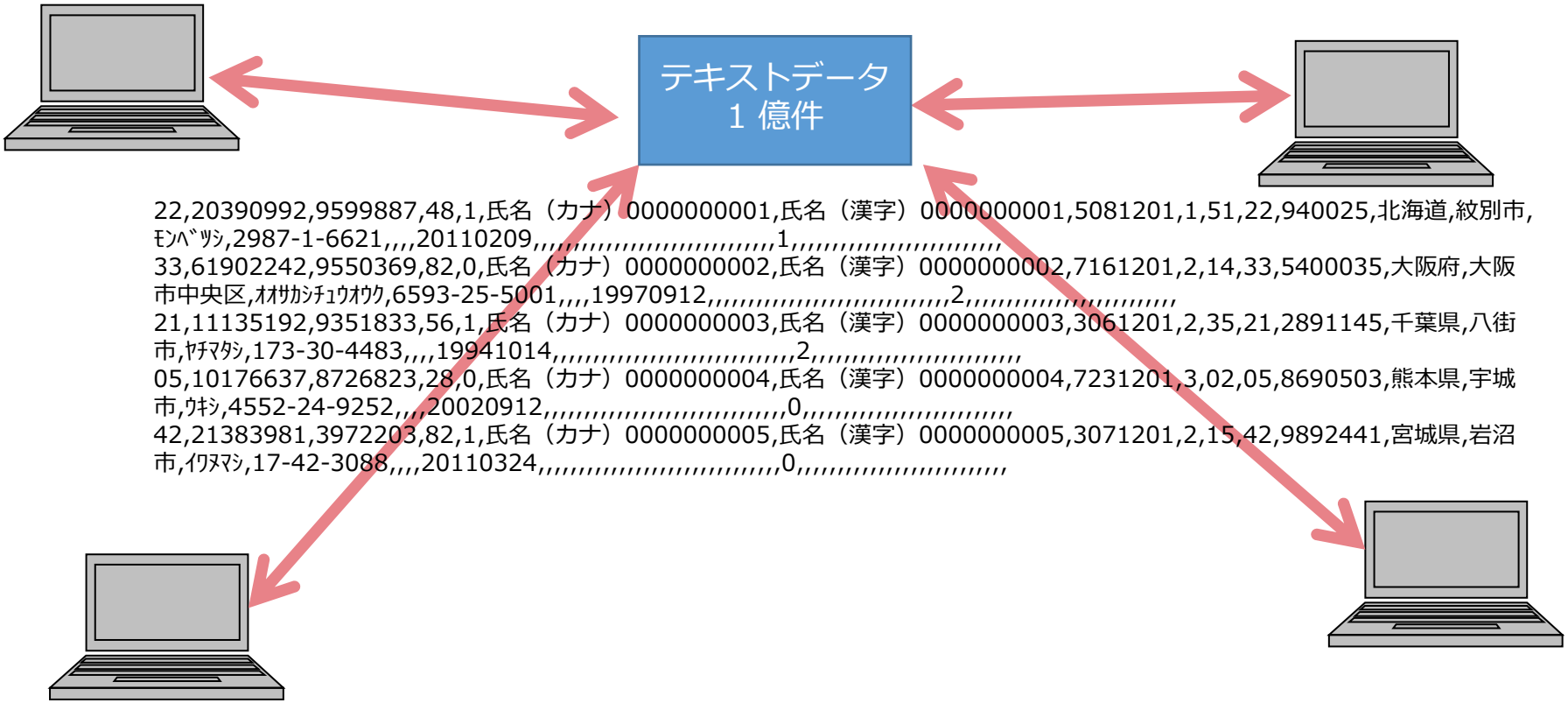
処理手順の概要

- 解析ブースターとSQL Serverのクエリー(Select文)は同一です。
- 同時実行数の制御にETLツールを用いています。

# 2.インメモリ処理

## 2-3. 件数 1億件、サイズ 20GBのテキストデータに対する文字列中間一致検索

検証シナリオ：単一のテキストデータファイルに対して文字列の中間一致検索を実行する。



# 2.インメモリ処理

## 2-3. 件数 1億件、サイズ 20GBのテキストデータに対する文字列中間一致検索

多重クエリー実行時間　　単位：ミリ秒

| 同時実行クエリー数 | 解析ブースター（インメモリ） | SQL Server |
|---|---|---|
| 1 | 2,544 | 11,078 |
| 2 | 2,618 | 22,222 |
| 3 | 2,730 | 33,400 |
| 4 | 3,056 | 44,470 |

参考値:　データロード時間　単位：ミリ秒

| 測定項目 | 解析ブースター（インメモリ） | SQL Server |
|---|---|---|
| ロード | 395,381 | 791,862 |

```
SELECT
項目01,項目02,項目03,項目04,
項目08,項目09,項目13,  --- 住所1
項目14,  --- 住所2
項目74,項目75,項目76
FROM SEARCH_ITEM
WHERE 項目13 like ‘%東京%’
and 項目14 like '%山%'
```

処理手順の概要

- 解析ブースターとSQL Serverのクエリー(Select文)は同一です。
- 同時実行数の制御にETLツールを用いています。

# 解析ブースター総括

## Kaiseki Booster is a Rapid Execution Tool

- EXCELやBIツールをはじめとする様々なアプリケーションとのハイレスポンスな連動が実現されます。

## Kaiseki Booster is a Rapid Development Tool

- SQL92ベースと豊富な関数および標準API が利用可能であり、アプリケーションの高速開発を可能にします。

## Kaiseki Booster promotes  reduction of T.C.O

- データベースが不要で省メモリ。他に類を見ない大量データのローコストオペレーションが可能になります。