Cloud Ace

大規模データをBigQueryで処理するときの課題とアイディアを少し

クラウドエース　宮城

大規模データを扱うときの BigQuery SQL実装における課題

1. スキャン量の削減
2. Resources Exceeded エラーの回避

今回の事例は、TrueData様の開発案件:「購買行動データ商用分析サービス ・Eagle Eye」です。

TrueData様と協力して開発し、この件で、True Data様にはNext19Tokyoに登壇頂きました。

Next19Tokyo

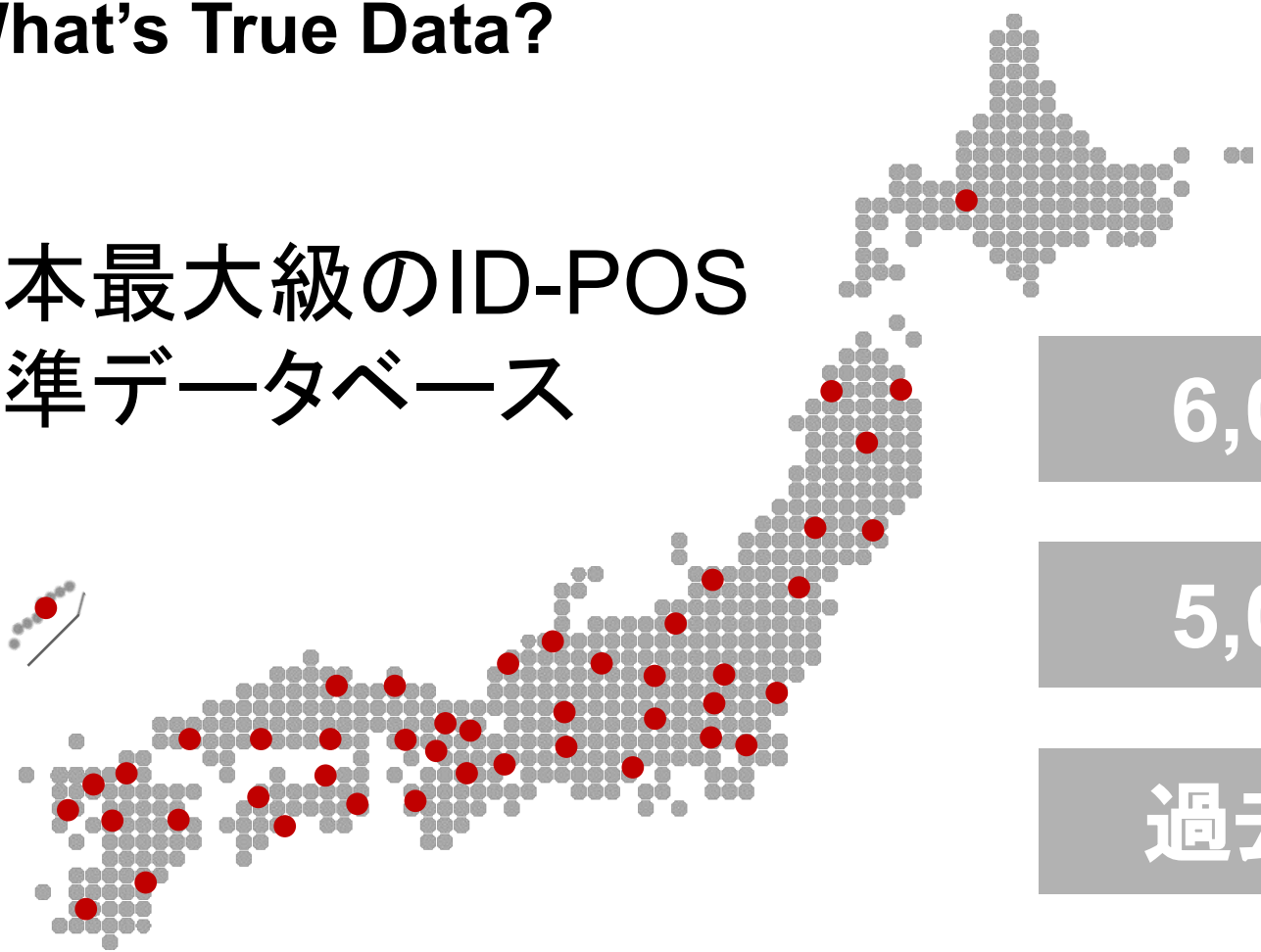
購買行動データ商用分析サービス
BigQuery活用事例 “Eagle Eye”

株式会社True Data
IT戦略室　池田 陽一
データマーケティング部　竹村博徳

What’s True Data?

True Data

日本最大級のID-POS標準データベース



6,000店舗

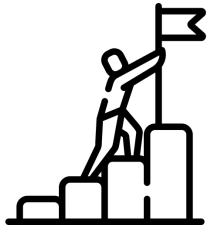
5,000万人

過去15年分

True Data IT Transformation

True Data

先頭を行こう



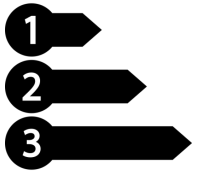
全商用分析サービスを
GCP、BigQueryへ
移行を決断。

システムの課題

増え続けるデータ量と処理量

**最大スケールに合わせた環境と
様々な制約**

優先順位



基幹サービス
Eagle Eyeから着手

2019年春移行完了

Why BigQuery?

True Data

理由

・環境制約からの脱却
・データ鮮度
・将来性

・スケールの極大化
・データ洗替が容易
・分析ならGoogle

懸念点

・料金
・くせ
・安定稼働

・従量課金
・シンプル
・独特なアーキテクチャー
・マネージドサービス

パートナー

クラウドエースと共に

・GCPの伝道師と二人三脚で
・BQ特性を活かしきった開発実施

How to solve concerns?

True Data

料金

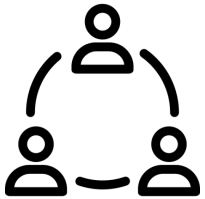
技術面:
BQ適合設計、実装
サービス面:
BQ適合メニュー提供

※**フラットレート未使用**

くせ

・**シャーディング**、
パーティショニングを正しく活用

安定稼働



・シェアしている環境
・**リトライ在り**きで処理、
サービス構築

Cloud Ace

スキャン量の削減

BigQuery においてスキャン量(＝コスト)削減は永遠のテーマ

スキャン量の削減

ストレージは安い。

US(マルチリージョン)では、

- アクティブストレージ : $0.020 per GB(1TBでも2200円)
- 長期保存(90日間変更なしで自動移行) : $0.010 per GB(1TBでも1100円)

東京リージョンでは、

- アクティブストレージ : $0.023 per GB(1TBでも2530円)
- 長期保存(90日間変更なしで自動移行) : $0.016 per GB(1TBでも1760円)

スキャン量の削減

クエリの料金、

US(マルチリージョン)では、

- オンデマンド：$5 per TB

東京リージョンでは、

- オンデマンド：$8.55 per TB

↑が、積み重なってくると結構な金額になる。

スキャンするデータ量で金額が決まるのでスキャン量をいかに減らすか、が課題。

スキャン量の削減

課題:

POSデータをまとめた売上データテーブルを使って分析したいが、

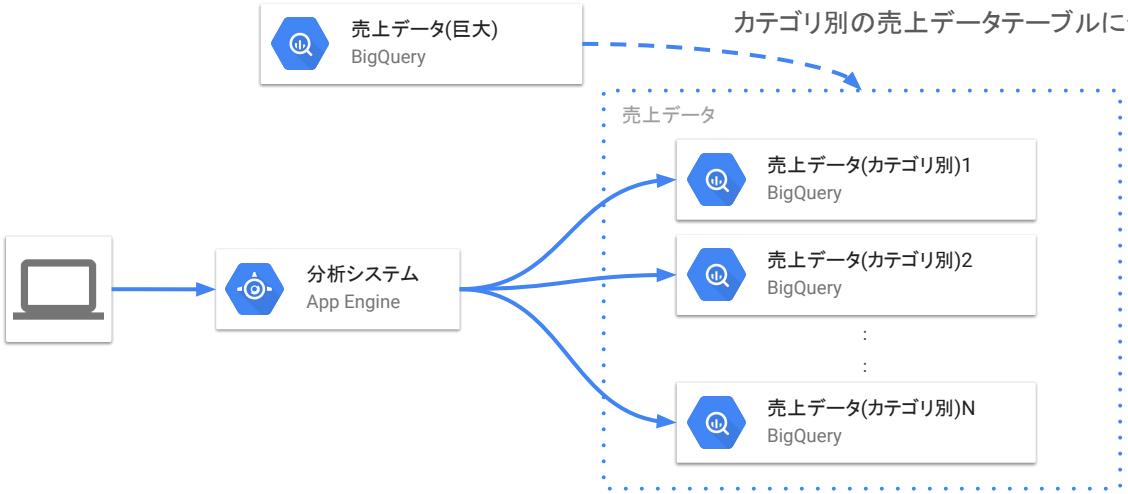
売上データテーブルは巨大でスキャン量もかかる。

(分割テーブルにしたとしても)

スキャン量の削減

対策:

売上データテーブルをカテゴリ(POSデータであれば肉、野菜など)ごとに分割する(日次バッチ実行)。

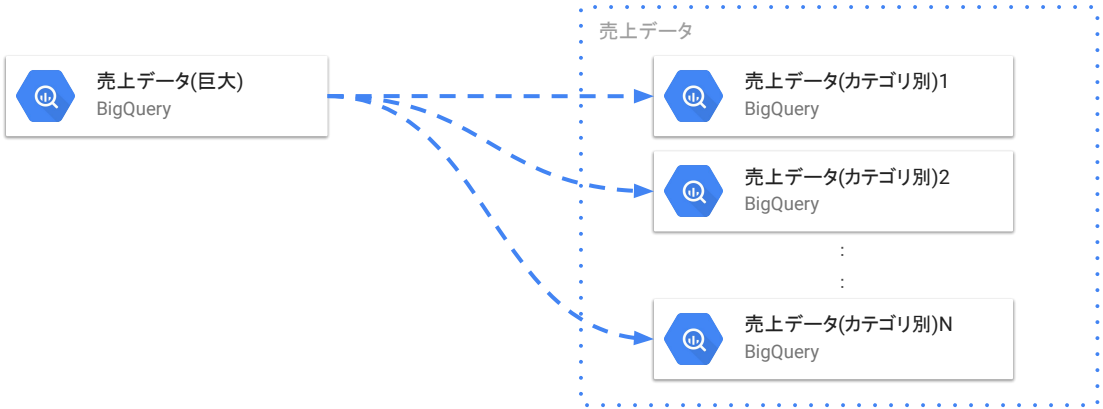


カテゴリ別に分けることで1回の分析のスキャン量を削減。

スキャン量の削減

ここでさらに問題が、、

カテゴリ別に分ける処理で売上データ(巨大)をフルスキャンすることになる。



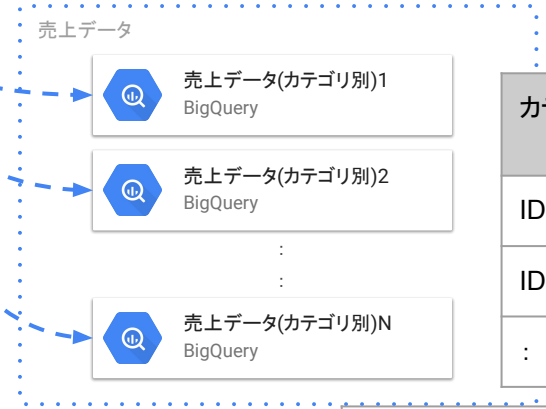
そこで、カテゴリ別の売上データテーブルに分ける処理を工夫しました。

スキャン量の削減

方法として、分割テーブルの Partition Field を応用する。

適当に設定した日付の `日付キー`(Partition Field) をカテゴリIDに紐づける

日付キー (Partition Field)	カテゴリID	売上日 (Not Partition Field)
2001-01-01	ID0001	2019-01-01
2001-01-02	ID0002	2019-01-01
:	:	:



カテゴリID	売上日 (Partition Field)
ID0001	2019-01-01
ID0002	2019-01-01
:	:

↑ あらかじめ売上データ(巨大)に Partition Field として日付キー(カテゴリIDと1対1)を入れておき、

カテゴリIDと紐づいている日付キーを WHERE句で指定して、特定のカテゴリIDの売上データのみを抽出 →

```
CREATE TABLE 売上データ(カテゴリ別)1
PARTITION BY 売上日
AS
SELECT *
FROM `売上データ(巨大)`
WHERE 日付キー = DATE("2001-01-01")
```

Resources Exceeded エラーの回避

分析が複雑になると SQL も複雑に(サブクエリの増加)。

データ量が少ないうちはエラーとならない SQL でも、データ量が増えるとResources Exceeded エラーが発生する。。。

ベストプラクティスとしては、サブクエリを実体化すること。

Resources Exceeded エラーの回避

分析が複雑になると SQL も複雑に(サブクエリの増加)。

データ量が少ないうちはエラーとならない SQL でも、データ量が増えるとResources Exceeded エラーが発生する。。。

ベストプラクティスとしては、サブクエリを実体化すること。

↓

そのようにします。。。

Window関数を使用するなど試行錯誤したんですが。。

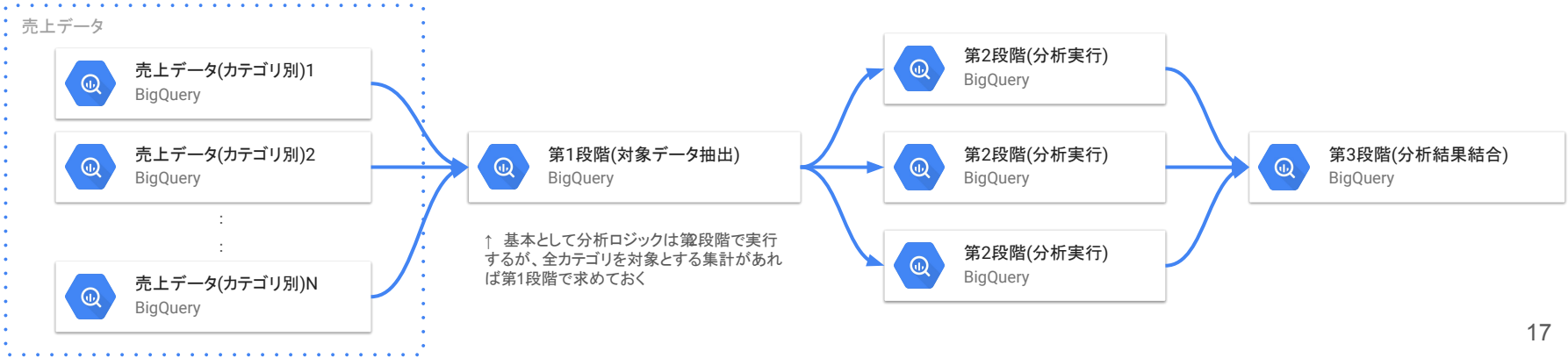
Resources Exceeded エラーの回避

分析クエリは3段階に分けて実体化。

第1段階：売上データから分析の条件(日付やカテゴリなど)を満たすデータを抽出(扱うデータを初期の段階でできるだけ減らす)

第2段階：抽出されたデータを使用して分析実行(テーブルは複数あり)

第3段階：分析結果テーブルを結合



Resources Exceeded エラーの回避

(補足)

3段階の分析クエリは Java & Task Queue にて制御。

BigQuery の気分次第でクエリが失敗するので、リトライは必須。

以上、ありがとうございました。