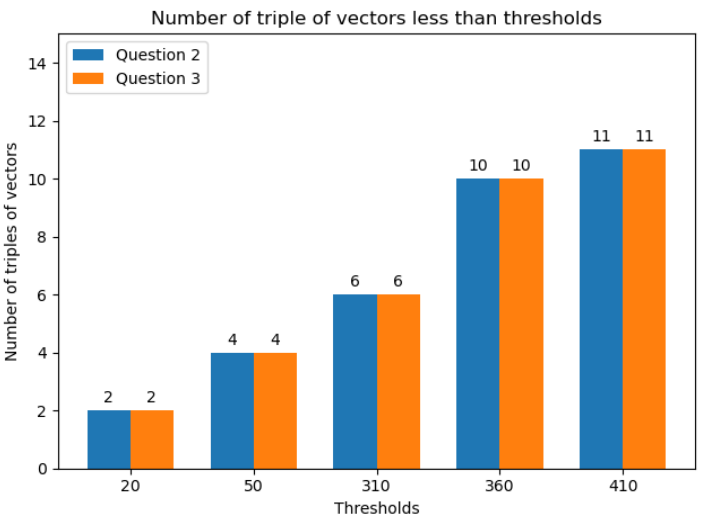
Technische Universiteit Eindhoven University of Technology

Discovery of Combinations of Vectors with Low Aggregate Variance

Q2: SQL query

```
SELECT id1, id2, id3, final_var
FROM (
    SELECT first(id1) as id1, first(id2) as id2, first(id3) as id3, sorted_id,
           SUM(var) + 2/10000*SUM(dot) - 2*SUM(mean_product) AS final_var
    FROM(
        SELECT t2.id1 as id1, t2.id2 as id2, t1.id as id3, t2.mean_product as mean_product,
            t1.var as var, t2.dot as dot,
            concat_ws('', array_sort(array(t2.id1, t2.id2, t1.id))) as sorted_id
        FROM (
            SELECT data.id as id,
                aggregate(data.int_list, cast(0 as long), (acc, x) -> acc + x*x)/size(data.
                    int_list) - POW(aggregate(data.int_list, 0, (acc, x) -> acc + x)/size(
                    data.int_list), 2) as var
            FROM data ) as t1
        CROSS JOIN (
            SELECT data1.id as id1, data2.id as id2,
                aggregate(zip_with(data1.int_list, data2.int_list, (x, y) -> x * y), 0, (acc
                    , x) -> acc + x) as dot,
                aggregate(data1.int_list, 0, (acc, x) -> acc + x)/size(data1.int_list) *
                    aggregate(data2.int_list, 0, (acc, x) -> acc + x)/size(data2.int_list)
                    as mean_product
            FROM data as data1, data as data2
            WHERE data1.id < data2.id
            ) as t2
        WHERE t1.id!= t2.id1 and t1.id!=t2.id2 )
    GROUP BY sorted_id )
WHERE final_var<=tau
```



$$\sigma^2 = (\frac{1}{l}\sum_{i=1}^{l}(X[i])^2) - (\sum_{j=1}^{l}\frac{X[j]}{l})^2 = \frac{1}{l}\sum_{i=1}^{l}(a_i+b_i+c_i)^2 - \mu^2$$

$$= \mathrm{var}(a) + \mathrm{var}(b) + \mathrm{var}(c) + \frac{1}{l}\sum_{i=1}^{l}(2a_ib_i + 2b_ic_i + 2a_ic_i) - 2(\overline{a}*\overline{b} + \overline{b}*\overline{c} + \overline{a}*\overline{c})$$

Q3: System structure

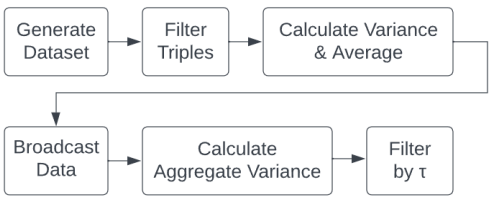
$rdd.keys() \xrightarrow{\text{cartesian}} keys2 \xrightarrow{\text{cartesian}} keys3$

$rdd \xrightarrow{\text{map}} var\_avg\_rdd \xrightarrow{\text{broadcast}} broadcast\_vectors$

$keys2 \xrightarrow{\text{map}} second\_term\_rdd \xrightarrow{\text{broadcast}} broadcast\_second\_term$

$var\_avg\_rdd \xrightarrow{\text{map}} var\_avg\_rdd2 \xrightarrow{\text{broadcast}} broadcast\_vectors$

$keys3 \xrightarrow{\text{map}} result \xrightarrow{\text{filter}} finalcount$



Optimization:

- Partition
- Cache
- Reduce computation size
- Broadcasting

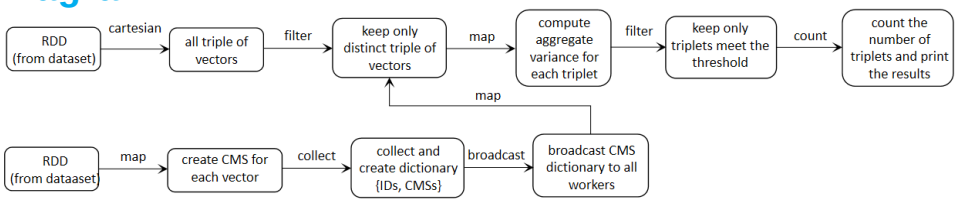
Joins and aggregations in SQL is not efficient, RDD performs better

Q4: Count-Min Sketch (CMS)

Theorem (Theorem 3 from Cormode et al. (2005)):

$$\mathbf{a}\odot\mathbf{a} \le \widehat{\mathbf{a}\odot\mathbf{a}}, \text{ and with probability } 1-\delta,\ \widehat{\mathbf{a}\odot\mathbf{a}} \le \mathbf{a}\odot\mathbf{a} + \frac{c\|\mathbf{a}\|_1\|\mathbf{a}\|_1}{l}$$

Diagram:



Pseudocode:

Algorithm 1 Estimate aggregate variance for each triple of vectors

```
function CMS(vector, ϵ, δ)
initialize 2D array count min sketch (CMS) with size ⌈ln(1/δ)⌉ × ⌈e/ϵ⌉ fill with zero
Hash each value into the cell of CMS
return CMS
end function

function AGGREGATE_VARIANCE(CMS_A, CMS_B, CMS_C, τ, ϵ)
CMS_agg ← CMS_A + CMS_B + CMS_C
CMS_aggInner ← inner_product(CMS_agg, CMS_agg)
aggregate variance ← min(CMS_aggInner)/10000 - mean(CMS_agg[0])²/10000
error ← τ + ϵ × ||CMS_agg[0]||₁ × ||CMS_agg[0]||₁/10000
return aggregate variance, error
end function

extract vector RDD IDS
triples of vectors with cartesian
keep distinct triples of vectors
create CMS for each vector: RDD.map(CMS(vector, ϵ, δ))
broadcast all CMSs to all workers
compute aggregate variance and error: AGGREGATE_VARIANCE(CMS_v1, CMS_v2, CMS_v3, τ, ϵ))
```

Results

	250 × 10000	1000 × 10000
Q2	37 s	timeout
Q3	19 s	7 min

Q4

ε	δ	τ	# triples	F1-score
0.001	0.1	<400	2572258	8.552×10^{-6}
0.01	0.1	<400	2562349	8.586×10^{-6}
0.0001	0.1	>200000	1	1.0
0.001	0.1	>200000	687	2.907×10^{-3}
0.002	0.1	>200000	742	2.690×10^{-3}
0.01	0.1	>200000	742	2.690×10^{-3}

Summary:

- F1-score is very close to 0.0 for almost all cases.
- ε=0.0001 and τ>200000, the number of triplets is same as exact result. However, CMS size for ε=0.0001 is greater than vector size.
- CMS is not useful for optimizing the functionalities and parameters in this context.

Q5: Ethical and other aspects

- For the company, the vector computation is not complex but the dataset can be too large for a relational database. Multi-threaded program on a powerful server also struggles with very large datasets.
- Spark on cloud resources is generally more scalable and costs less than Spark on a company-owned cluster. However, storing sensitive data on a cloud-based platform can raise data privacy issues. If speed is more important than accuracy, then the company should choose the approximation technique on cloud resources.
- Random sampling can be used to approximate the estimation of aggregate variance for triple of vectors. To improve accuracy of the estimate, we can repeat random sampling process multiple times and average the results. This techniques can be applied on powerful server and on cloud resourses.
- Dimensionality reduction can be used to reduce the dimensionality of vectors. This can make it possible to compute the vector combinations on a smaller size. This techniques can be applied on powerful server and on cloud resourses.