Short Paper

Translation Grids for Multi-way Join Size Estimation

Magnus Müller
magnus.mueller@uni-mannheim.de
University of Mannheim
Mannheim, Germany

Guido Moerkotte
guido.moerkotte@uni-mannheim.de
University of Mannheim
Mannheim, Germany

ABSTRACT

We present a novel approach to estimate query result sizes for queries containing multiple joins. Our approach relies on (1) enhanced AKMV sketches and (2) a novel data structure called translation grid. In essence, we obtain estimates by connecting hashes from AKMV sketches via a translation grid.

1 INTRODUCTION

In query optimization, *cardinality estimates*, i.e., estimates for query result sizes, play an important role [8]. For instance, given a query for which an execution plan must be found, cardinality estimates are used to decide on the join order and whether to re-partition or broadcast tables in a distributed hash join.

In this paper, the **problem we solve** is to estimate the result cardinality of queries containing multiple joins. Consider, for instance, a star query in which the two dimension tables `D1, D2` join with the fact table `F`:

```
SELECT count(*) FROM D1 JOIN F ON D1.A = F.A JOIN D2
ON D2.B = F.B AND D1.X < 5 AND D2.Y != "foo"
```

Our approach to derive estimates relies on two data structures: (1) Well established AKMV sketches [2], which we present in Section 2 and enhance with new operations, including operations for two-way joins, in Section 3. As well as (2) the *translation grid* (Section 4), a novel data structure that helps to estimate the size of multi-way joins. To motivate translation grids, observe that in the above example query, the joint frequency distribution between `F.A` and `F.B` influences the size of the three-way join. Here, translation grids come into play. Translation grids allow to determine attribute value pairs that exist, with non-zero frequency, in a joint frequency distribution. We refer to this existence criterion as *joint existence distribution* (JED). However, instead of operating on actual attribute values, translation grids operate on hashes. In particular, translation grids track pairs of hashes that exist, or might exist, in a pair of AKMV sketches.

2 PRELIMINARIES

This section presents preliminaries on which we build in later sections. In particular, we introduce the established KMV and AKMV sketches. As part of the following discussion, we use relations R and T with attribute sets C and D, respectively.

The *k minimum value sketch* (KMV) [1] is a synopsis that allows to estimate the number of distinct values (NODV) in a multiset. In our context, the multiset under consideration consists of the values in $R.C$. A KMV is constructed in a single pass over $R.C$ and requires only a constant amount of memory. The basic idea is simple: To construct a KMV, hash each entry in $R.C$ to $[0, 1]$ and keep track of the k smallest hashes, where k is some fixed number. We refer to the tracked hashes by the totally ordered set $\mathcal{H} := \{h_1 < h_2 < \cdots < h_k\}$. Then, to estimate the NODV in $R.C$, use h_k as an indicator. In particular, a large h_k indicates few distinct entries in $R.C$ and, vice versa, a small h_k indicates many distinct input entries in $R.C$. As a formula, the KMV estimate for the NODV is

$$\widehat{d}_{KMV} = \begin{cases} k/h_k & \text{, if } d > k \\ k & \text{, else} \end{cases} \tag{1}$$

where $d := |R.C|_d$, and throughout the paper we use $|x|_d$ for the number of distinct entries in expression x. As the formula suggests, for fewer than k distinct values, the estimate is the true value. For simplicity, throughout this paper, we use only one deterministic global hash function $H : \circ \to [0, 1]$ that is capable of hashing any input to $[0, 1]$. To simplify the exposition, we always assume the common case $d > k$. In this paper, unless explicitly stated otherwise, all KMVs/AKMVs share the same fixed parameter k.

The *augmented KMV* (AKMV) by Beyer et al. [2] is an extension of KMV. The main idea of AKMV is to augment the KMV by counters to track the multiplicity by which each hash is seen during construction. The counters enable one to estimate the NODV in a multiset that is the intersection, union, or difference of other multisets. For instance, three AKMVs can be used to estimate the NODV in the intersection of three multisets. In the following, we formally define AKMVs, introduce the multiset operations they support, and show how AKMVs are used for NODV estimation.

We formally define an AKMV for $R.C$ as $\mathcal{S}_{R.C} := (\mathcal{H}_{R.C}, \eta_{R.C})$, where $\mathcal{H}_{R.C}$ denotes the set of the k smallest hashes we know from KMV, and the function $\eta_{R.C} : [0, 1] \to \mathbb{N}$ returns the tracked multiplicity of h if $h \in \mathcal{H}_{R.C}$. Otherwise, if $h \notin \mathcal{H}_{R.C}$, we define $\eta_{R.C}(h) = 0$.

AKMVs support the multiset operations union $\cup$, intersection $\cap$, and multiset difference $\setminus$. Let $\mathcal{S}_{R.C}, \mathcal{S}_{T.D}$ be two AKMVs. The result of $\mathcal{S}_{R.C} \circ \mathcal{S}_{T.D}$, where $\circ \in \{\cup, \cap, \setminus\}$, is a new AKMV $\mathcal{S}_{R.C \circ T.D}$. For brevity, let $E := R.C \circ T.D$. The set of hashes $\mathcal{H}_E$ in $\mathcal{S}_E$ is defined as the k smallest hashes in $\mathcal{H}_{R.C} \cup \mathcal{H}_{T.D}$ and each $h \in \mathcal{H}_E$ has multiplicity

$$\eta_E(h) = \begin{cases} \eta_{R.C}(h) + \eta_{T.D}(h) & \text{, if } \circ = \cup \\ \min(\eta_{R.C}(h), \eta_{T.D}(h)) & \text{, if } \circ = \cap \\ \max(\eta_{R.C}(h) - \eta_{T.D}(h), 0) & \text{, if } \circ = \setminus \end{cases} \tag{2}$$

Note that intersect and subtract operations can cause hashes with a multiplicity of zero. In case of intersection, this indicates that some entry was present only in one of $R.C$ and $T.D$.

The appropriate NODV estimate for an arbitrary AKMV $\mathcal{S}$ must reflect the number of hashes with multiplicity greater zero, to which we refer as $p := |\{h_i \in \mathbf{H} \mid \eta(h_i) > 0\}|$. Equation 1 is not applicable for AKMVs, since p is not reflected. Instead, the formula to compute the NODV estimate $\widehat{d}_{AKMV}$ of $\mathcal{S}$ is

$$\widehat{d}_{AKMV} = \frac{p}{k} \cdot \frac{k-1}{\max(\mathcal{H})} \tag{3}$$

In the formula, the first fraction is the ratio of tuples with multiplicity greater zero. The second fraction is almost the KMV

estimate from Equation 1 - Beyer et al. have shown that decrementing the numerator by 1 is necessary to make the estimator unbiased [2].

Example: Consider AKMV $\mathcal{S} = (\mathcal{H}, \eta) = (\{0.0002 < 0.003 < 0.008 < 0.015\}, (30, 50, 0, 40))$. Observe that $\max(\mathcal{H})$=0.015 and p=3 hashes have multiplicity greater zero. Hence, assuming k=4, the NODV estimate $\widehat{d}_{AKMV}$ of $\mathcal{S}$ is $3/4 \cdot (4-1)/0.015 = 150$.

3 NEW OPERATIONS FOR AKMV SKETCHES

This section presents two new estimators and one new operation for AKMVs.

For an AKMV $\mathcal{S}_E$ constructed on some expression E, we define its *distinct ratio* as the fraction of tracked hashes, i.e. $k/|E|_d$. We say that an AKMV with distinct ratio x is an x-AKMV. In Section 4.4, we apply the distinct ratio as a scaling factor. The following lemma shows how to estimate the distinct ratio of $\mathcal{S}_E$.

Lemma 3.1. *An estimate for the distinct ratio of some AKMV $\mathcal{S}_E$ is $\frac{p_E}{\widehat{d}_{AKMV}}$.*

The rationale behind the estimate is simple. As mentioned above, when $\mathcal{S}_E$ was constructed on E, its distinct ratio is $k/|E|_d$ and we simply substitute $|E|_d$ by its estimate $\widehat{d}_{AKMV}$. The numerator reflects cases where $\mathcal{S}_E$ is the result of a multiset operation of two AKMVs. Then, some hashes in $\mathcal{H}_E$ might have multiplicity 0. To reflect that hashes with multiplicity zero have no corresponding attribute values in E, we use p_E in the numerator.

Next, we define a multiplication operation for AKMV sketches. The result of multiplying two AKMVs $\mathcal{S}_{R.C}, \mathcal{S}_{T.D}$ gives an AKMV for the expression $\pi_C(R \bowtie_{C=D} T)$, or, equivalently, $\pi_D(R \bowtie_{C=D} T)$.

Definition 3.2. Let $\mathcal{S}_{R.C}, \mathcal{S}_{T.D}$ be two AKMVs. Their product $\mathcal{S}_{R.C} \cdot \mathcal{S}_{T.D}$ gives a new AKMV $\mathcal{S}_E$, where $\mathcal{H}_E$ contains the k smallest hashes in $\mathcal{H}_{R.C} \cup \mathcal{H}_{T.D}$ and the multiplicity of each $h \in \mathcal{H}_E$ is $\eta_E(h) = \eta_{R.C}(h) \cdot \eta_{T.D}(h)$.

AKMV multiplication allows us to derive AKMVs for join results from base table AKMVs. However, note that the derived AKMV $\mathcal{S}_E$ is not necessarily equal to an AKMV constructed on $\pi_C(R \bowtie_{C=D} T)$, since only $\mathcal{S}_E$ might contain hashes with multiplicity zero.

Next, we observe that, in addition to NODV estimation, an AKMV $\mathcal{S}_E$ can be used to estimate the cardinality $c := |E|$, i.e., the number of not necessarily distinct values in E.

Lemma 3.3 (cardinality estimate). *Let $M_E := \sum_{h \in \mathcal{H}_E} \eta_E(h)$ be the summed multiplicities of the hashes in $\mathcal{H}_E$ of AKMV $\mathcal{S}_E$. An estimate for the cardinality c of relational algebra expression E is*

$$\widehat{c}_{AKMV} := \frac{M_E}{k} \cdot \frac{k-1}{\max(\mathcal{H}_E)}$$

Note that $\widehat{c}_{AKMV}$ is the simplified expression one receives from scaling $\widehat{d}_{AKMV}$ by the average multiplicity of the hashes with non-zero multiplicity in $\mathcal{H}_E$. The estimate $\widehat{c}_{AKMV}$ is unbiased, in the sense that $\mathbf{E}[\widehat{c}_{AKMV}] = c$, since hashes with small multiplicities have the same probability of being included in $\mathcal{H}_E$ as hashes with large multiplicities. In particular, according to Beyer et al. [2], the hashes in $\mathcal{H}_E$ constitute a random sample of the hashed values in E.

Corollary 3.4. *$\widehat{c}_{AKMV}$ of $\mathcal{S}_{R.C} \cdot \mathcal{S}_{T.D}$ is an unbiased estimate for $|\pi_C(R \bowtie_{C=D} T)|$*

Algorithm 1 Translation grid construction.

```
Construct(R, A, B, m, n, hA_max, hB_max)
   Input:  Relation R,
           attribute sets A, B of R
           two numbers m, n
           largest hash values hA_max, hB_max in AKMVs of R.A and R.B
   Output: Translation grid T(R.A, R.B)
1  Let T be a translation grid of m × n tiles
2  for t ∈ R
3      hA = H(t.A)
4      hB = H(t.B)
5      if hA > hA_max or hB > hB_max: continue
6      tile = T.tile_ref(hA, hB)
7      tile.B_A.insert(hA)
8      tile.B_B.insert(hB)
9  return T
```

Hence, the size of an equi-join can be estimated from two AKMVs constructed for the join attributes in the base tables. Note that $|\pi_C(R \bowtie_{C=D} T)| = |\pi_D(R \bowtie_{C=D} T)| = |R \bowtie_{C=D} T|$.

Example: Suppose $\mathcal{S}_{R.C} = (\{0.0002 < 0.003 < 0.015 < 0.05\}, (3, 5, 4, 8))$ and $\mathcal{S}_{T.D} = (\{0.0002 < 0.003 < 0.008 < 0.015\}, (10, 10, 10, 10))$. The product of $\mathcal{S}_{R.C} \cdot \mathcal{S}_{T.D}$ is $\mathcal{S}_E = (\{0.0002 < 0.003 < 0.008 < 0.015\}, (30, 50, 0, 40))$. Since $M_E = 120$ is the sum of the multiplicities, the join size $|R \bowtie_{C=D} T|$ is estimated as $\widehat{c}_{AKMV} = 120/4 \cdot (4-1)/0.015 = 6000$.

4 TRANSLATION GRIDS

This section presents the *translation grid*, a data structure that approximates the JED, cf. Section 1, of a pair of attributes/ attribute sets. Translation grids, like AKMVs, entirely operate on the hashes of attribute values. We use translation grids to connect hashes from two AKMVs.

The example query from the introduction describes a situation where translation grids are useful. Given AKMVs for each column $D1.A$, $D2.B$, $F.A$, $F.B$, we can apply AKMV multiplication to obtain AKMVs $\mathcal{S}_{D1.A \bowtie F.A}$, $\mathcal{S}_{D2.B \bowtie F.B}$ for the two-way joins. Then, the translation grid connects hashes in $\mathcal{S}_{D1.A \bowtie F.A}$ with hashes in $\mathcal{S}_{D2.B \bowtie F.B}$ such that the JED between $F.A$ and $F.B$ is captured.

4.1 Structure

A translation grid $\mathcal{T}(R.A, R.B)$ is a two-dimensional grid of $m \times n$ *tiles*. For each tuple $t \in R$, the pair of hashes $(H(t.A), H(t.B))$, where H is the same hash function as for AKMV, maps to one tile. Each tile maintains two Bloom filters $\mathcal{B}_{R.A}, \mathcal{B}_{R.B}$. All $(H(t.A), H(t.B))$ corresponding to the same tile are inserted into the respective Bloom filters $\mathcal{B}_{R.A}, \mathcal{B}_{R.B}$ of that tile. As we will see, the fact that we insert hashes into the Bloom filters of the tiles of translation grid $\mathcal{T}(R.A, R.B)$, allows us to test whether a hash pair (h_A, h_B) from two AKMVs $\mathcal{S}_{R.A}, \mathcal{S}_{R.B}$ corresponds to attribute values in R.

4.2 Construction

The construction of translation grid $\mathcal{T}(R.A, R.B)$ happens in a single run over relation R. The construction of $\mathcal{T}(R.A, R.B)$ takes place *after* the construction of AKMVs $\mathcal{S}_{R.A}, \mathcal{S}_{R.B}$. Algorithm 1 describes the translation grid construction process. For an illustration see Figure 1. The arguments are relation R, with attribute sets A and B, and the desired dimensions of $\mathcal{T}(R.A, R.B)$ specfied by m and n. In addition, the parameters hA_{max} and hB_{max} are the largest hash values found in any AKMV of $S.A$ and $S.B$, respectively. In line 1, $\mathcal{T}(R.A, R.B)$ is initialized with the bits of the

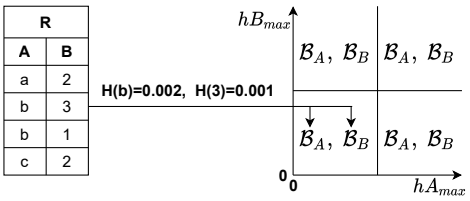

Figure 1: Construction of translation grid.

two Bloom filters of each tile set to zero. Then, we iterate over each tuple $t \in R$ and hash each $t.A$ and $t.B$. In line 5, all hash pairs (hA, hB) with $hA > hA_{max} \vee hB > hB_{max}$ are discarded, i.e., we discard all hash combinations that are definitely not tracked by AKMVs. Filtering unnecessary (hA, hB) values helps to significantly reduce the size of a translation grid, or, equivalently, increase the accuracy it delivers for a given space consumption. For the remaining hash pairs (hA, hB), the corresponding tile is identified, and each hash is inserted in its respective Bloom filter. Finally, $\mathcal{T}(R.A, R.B)$ is returned.

4.3 Translation

By *translating*, we denote the process of finding combinations of hashes from two input AKMVs $\mathcal{S}_{R.A}, \mathcal{S}_{R.B}$ that $\mathcal{T}(R.A, R.B)$ has seen during construction. All qualifying combinations of hashes are output in two new AKMVs $\mathcal{S}'_{R.A}, \mathcal{S}'_{R.B}$. Note that, in the context of multi-way join size estimation, the input AKMVs $\mathcal{S}_{R.A}, \mathcal{S}_{R.B}$ are each the result of an intersect or multiply operation with another AKMV.

Algorithm 2 describes how to translate two input AKMVs using a translation grid. For each combination of hashes (hA, hB) with multiplicity greater zero, i.e., $\eta_{R.A}(hA) > 0 \wedge \eta_{R.B}(hB) > 0$, the corresponding tile is identified in line 4. Then, in line 5, we test if hA exists in $\mathcal{B}_A$ and if hB exists in $\mathcal{B}_B$ of the identified tile. Note that, as always with Bloom filters, false positives are possible. All combinations of hashes (hA, hB) that pass the test are tracked in the two output AKMVs $\mathcal{S}'_{R.A}, \mathcal{S}'_{R.B}$. Essentially, the output AKMVs $\mathcal{S}'_{R.A}, \mathcal{S}'_{R.B}$, as returned in line 8, are filtered versions of $\mathcal{S}_{R.A}, \mathcal{S}_{R.B}$. As we will see in the next section, $\mathcal{S}'_{R.A}, \mathcal{S}'_{R.B}$ are proper AKMVs that can be used for cardinality estimation.

4.4 Estimation Properties of Translation grids

This section presents two Lemmata that describe how translation grids can be used, in conjunction with AKMVs, for cardinality estimation.

First, we define an ideal translation grid to which we refer as a *lossless translation grid.*

Definition 4.1 (Lossless translation grid). A lossless translation grid $\mathcal{T}(R.A, R.B)$, correctly determines for each $h_A \in \mathcal{S}_{R.A}, h_B \in \mathcal{S}_{R.B}$ iff (h_A, h_B) corresponds to a tuple in R. There are no false-positives in a lossless translation grid.

A lossy translation grid is one that is not lossless (produces false-positives).

In the following lemma, we combine a lossless translation grid together with two AKMVs. To understand the Lemma, recall that by a x-AKMV we refer to an AKMV that tracks a fraction x of the hashes seen during construction.

Lemma 4.2. *Let $\mathcal{T}(R.A, R.B)$ be a lossless translation grid of relation R with $|R|=|R.A|_d=|R.B|_d$. When a x-AKMV $\mathcal{S}_{R.A}$, $x \in [0, 1]$, and a 1.0-AKMV $\mathcal{S}_{R.B}$ are translated by $\mathcal{T}(R.A, R.B)$, then each hash $h \in \mathcal{S}_{R.A}$ necessarily finds its correct matches in $\mathcal{T}(R.A, R.B)$ (but no false positive). For the output AKMV $\mathcal{S}'_{R.A}$, it follows that $\mathbf{E}[\widehat{d}_{AKMV}(\mathcal{S}'_{R.A})] = \mathbf{E}[\widehat{c}_{AKMV}(\mathcal{S}'_{R.A})] = |R|$.*

Algorithm 2 Translation grid, translation of AKMVs.

```
1 TRANSLATE(T, S_R.A, S_R.B)
    Input:  Translation grid T,
            two input AKMVs S_R.A, S_R.B
    Output: Two output AKMVs S'_R.A, S'_R.B
2   Let S'_R.A, S'_R.B be two empty AKMVs
3   for hA ∈ S_R.A with η_R.A(hA) > 0
4       for hB ∈ S_R.B with η_R.B(hB) > 0
5           tile = T.TILE_REF(hA, hB)
6           if tile.B_A.CONTAINS(hA) and tile.B_B.CONTAINS(hB)
7               S'_R.A.INSERT(hA)
8               S'_R.B.INSERT(hB)
9   return S'_R.A, S'_R.B
```

Essentially, what Lemma 4.2 states is that, in the above setting, $\mathcal{S}_{R.A} = \mathcal{S}'_{R.A}$. The same is not true for $\mathcal{S}'_{R.B}$, though. However, we clearly do not want to rely on 1.0-AKMVs. Luckily, the following lemma comes to the rescue.

Lemma 4.3. *Suppose the same setting as in Lemma 4.2, but this time $\mathcal{S}_{R.B}$ is a y-AKMV, where $y \in [0, 1]$. In expectation, each $h \in \mathcal{S}_{R.A}$ finds a ratio of y of its correct matches in $\mathcal{T}(R.A, R.B)$ (but no false positive). It follows that $\mathbf{E}\left[\frac{\widehat{c}_{AKMV}(\mathcal{S}'_{R.A})}{y}\right] = |R|$.*

Note that, from symmetry, also $\mathbf{E}\left[\frac{\widehat{c}_{AKMV}(\mathcal{S}'_{R.B})}{x}\right] = |R|$ holds. In our evaluation, we present an experiment to illustrate Lemma 4.3.

Observation: Lemma 4.3 is applicable for multi-way join size estimation. For instance, let $E1, E2$ be relational expressions with key attributes A, B, respective, and $\mathcal{T}(F.A, F.B)$ a translation grid for a relation F with foreign key attributes A, B. Assume AKMVs $\mathcal{S}_{E1.A}, \mathcal{S}_{F.A}, \mathcal{S}_{F.B}, \mathcal{S}_{E2.B}$ exist. Applying Lemma 4.3 to the output AKMVs of TRANSLATE$(\mathcal{T}(F.A, F.B),\ \mathcal{S}_{F.A} \cdot \mathcal{S}_{E1.A},\ \mathcal{S}_{F.B} \cdot \mathcal{S}_{E2.B})$ gives an estimate for $|E1 \bowtie_{E1.A=F.A} F \bowtie_{F.B=E2.B} E2|$. Section 5 presents an experiment on multi-way join size estimation.

5 EVALUATION

In our evaluation, we present two experiments. The first experiment demonstrates the validity and precision of Lemma 4.3. In the second experiment, we estimate result cardinalities of three-way joins.

Experiment 1: Let R: [{A, B}] with $|R|=|R.A|_d=|R.B|_d$=1000. We construct $\mathcal{S}_{R.A}$ as 0.3-AKMV and $\mathcal{S}_{R.B}$ is a y-AKMV as specified in Table 1. The lossless translation grid $\mathcal{T}(R.A, R.B)$ is approximated by $|R.A|_d \times |R.B|_d$ tiles. We denote $cA := \widehat{c}_{AKMV}(\mathcal{S}'_{R.A})$ and $cB := \widehat{c}_{AKMV}(\mathcal{S}'_{R.B})$. Clearly, if Lemma 4.3 holds, we expect $cA/y \approx cB/0.3 \approx 1000$. Table 1 shows the result. Indeed, even for y=0.1, we have $cA/y \approx 1000$. For comparison, we include cA, cB. Note how cA decreases as y decreases, since each $h \in \mathcal{S}'_{R.A}$ finds fewer and fewer matches in $\mathcal{T}(R.A, R.B)$.

How many hash pairs (hA, hB) are inserted into $\mathcal{T}(R.A, R.B)$ during construction? By line 5 in Algorithm 1, only hashes possibly tracked by AKMVs are inserted into $\mathcal{T}(R.A, R.B)$. Hence, we expect $0.3 \cdot y \cdot |R|$ many inserts, cf. the last column of Table 1. The second to last column shows the actual number of inserts. Observe that the actual number is both small and close to the expected number.

Experiment 2: Recall the star query from the introduction. For varying selection predicates, we estimate its result cardinality, i.e., $|\sigma_{p_{D1}}(D1) \bowtie_{D1.A=F.A} F \bowtie_{F.B=D2.B} \sigma_{p_{D2}}(D2)|$. The dimension tables have $|D1|$=1000 and $|D2|$=2000 rows, respectively,

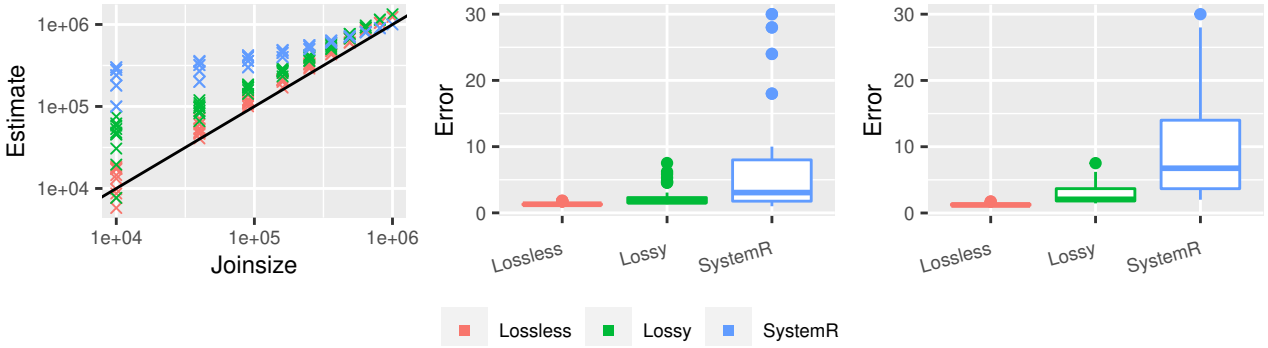

Figure 2: Results of Experiment 2: Scatterplot (left), errors over all queries (middle), errors over selective queries (right).

Table 1: Results of Exper. 1. As expected $\frac{cA}{y} \approx \frac{cB}{x} \approx 1000$.

y	cA	cA/y	cB	$cB/0.3$	ins.	exp. ins.
1	1075.48	1075.48	300.471	1001.57	300	300
0.9	949.591	1055.1	291.39	971.3	265	270
0.8	838.086	1047.61	295.724	985.746	234	240
0.7	715.79	1022.56	288.319	961.063	200	210
0.6	600.688	1001.15	287.529	958.431	168	180
0.5	504.093	1008.19	288.892	962.974	141	150
0.4	414.077	1035.19	297.449	991.497	116	120
0.3	316.527	1055.09	296.691	988.969	87	90
0.2	209.791	1048.95	317.068	1056.89	58	60
0.1	111.541	1115.41	326.231	1087.44	30	30

and the attributes A, B refer to their respective primary key. The fact table $F := \{[i,j] | i \in [0, |D1|), j \in [0, |D2|), \frac{i}{|D1|} \geq \frac{j}{|D2|}\}$ has the skewed foreign key attributes $F.A, F.B$ that are correlated with each other. Its size is $|F|$=1, 000, 000. The cardinality of a query result is influenced by the selection predicates $p_{D1} \equiv D1.A < c_1$, $p_{D2} \equiv D2.B \geq c_2$, where we compute results for all combinations with non-zero cardinality of $c_1 \in \{100, 200, \ldots, 1000\}$, $c_2 \in \{0, 200, 400, \ldots, 1800\}$. Hence, the true cardinality varies between 10k and 1M. We make the following assumption: Since $D1, D2$ are small, AKMVs $\mathcal{S}_{\pi_r(\sigma_{p_{D1}}(D1))}, \mathcal{S}_{\pi_t(\sigma_{p_{D2}}(D2))}$ can be built on the fly. Since F is large, $\mathcal{S}_{F.A}, \mathcal{S}_{F.B}$ and $\mathcal{T}(F.A, F.B)$ must be computed offline, i.e. not per query. All AKMVs are of size k=100. An estimate for the query result size is obtained in three steps: (1) Compute $\mathcal{S}_{J1} := \mathcal{S}_{F.A} \cdot \mathcal{S}_{\pi_r(\sigma_{p_{D1}}(D1))}$ and $\mathcal{S}_{J2} := \mathcal{S}_{F.B} \cdot \mathcal{S}_{\pi_t(\sigma_{p_{D2}}(D2))}$, (2) Obtain $(\mathcal{S}'_{J1}, \mathcal{S}'_{J2}) = \textsc{Translate}(\mathcal{T}(F.A, F.B), \mathcal{S}_{J1}, \mathcal{S}_{J2})$, and (3) estimate the cardinality as $\widehat{c}_{AKMV}(\mathcal{S}'_{J1})/\widehat{r}$, where $\widehat{r}$ is the estimated distinct ratio of $\mathcal{S}'_{J2}$, cf. Lemma 3.1. In Figure 2, for each query, we show three different cardinality estimates. (1) An estimate obtained by a lossless translation grid, which tracks only 5200 $\approx$ 0.5% $\cdot$ $|F|$ pairs of hashes. The size of only $5200 \cdot 2 \cdot 4B = 41.6KB$ demonstrates the practicability of lossless translation grids. (2) A lossy 100 $\times$ 100 translation grid, in which only 2600 tiles are actually used. Each tile contains only two 32-bit Bloom filters. Hence, the used tiles consume only $2600 \cdot 2 \cdot 4B = 20.8KB$, corresponding to only 0.26% of F's memory footprint, assuming each tuple in F consists of 2 integers of 32-bit. (3) For comparison, we show the System R [12] estimator, as implemented in Postgres version 13.2. In Figure 2, the scatterplot on the left shows the correlation between the observed and estimated cardinality. The best estimates are obtained using the lossless translation grid, the lossy translation grid produces slightly worse estimates. System R significantly overestimates, since its assumptions do not hold. The boxplots in the middle of Figure 2, illustrate the estimation errors for each estimator from the scatterplot. The estimation error is measured by the q-error [9], defined as $\max(\frac{\widehat{x}}{x}, \frac{x}{\widehat{x}})$ for a value x and its estimate $\widehat{x}$. The boxplots on the right are restricted to only the challenging queries with selectivity of p_{D1} less than 0.5. Note how the errors significantly worsen for System R when looking only at the more selective queries. The estimates by the lossless and lossy translation grid are less affected.

6 RELATED WORK

The most notable related work is a recent paper by Izenov et al. [6]. Just as we do, Izenov et al. focus on multi-way join size estimation. However, instead of AKMV sketches and translation grids, they rely on fast-AGMS sketches and a technique they call *sketch merging*.

The concept of JED, introduced in Section 1, is related to frequency matrices [5, 11].

The earliest approach to join size estimation, and cardinality estimation in general, is System R [12]. More recent approaches rely on sampling [3, 13] or machine learning [7].

Using AKMVs in conjunction with bucket sketches [10], where the AKMVs replace HyperLogLogs [4] as the underlying sketch, allow us to obtain AKMVs for a subset of some relation *after* AKMVs were constructed on the complete table.

7 CONCLUSION

We presented a novel approach to estimate query result sizes for queries containing multiple joins. Our approach relies on two novelties. (1) New operations for AKMV sketches, in particular distinct ratio estimation, AKMV multiplication for two-way join approximation, and cardinality estimation. (2) A novel data structure called *translation grid* that partitions the hash space of AKMVs and tracks combinations of hashes from two AKMVs in such a way that they approximate the joint existence distribution of two attributes/attribute sets. In our evaluation, we demonstrated that our approach allows to estimate the cardinality of three-way joins. For future work, we hope to consider even more joins. Here, similar to sampling-based approaches, a major challenge is to prevent AKMVs from having a multiplicity of zero for all tracked hashes.

REFERENCES

[1] Ziv Bar-Yossef, TS Jayram, Ravi Kumar, D Sivakumar, and Luca Trevisan. 2002. Counting distinct elements in a data stream. In *International Workshop on Randomization and Approximation Techniques in Computer Science*. Springer, 1–10.

[2] Kevin Beyer, Peter J Haas, Berthold Reinwald, Yannis Sismanis, and Rainer Gemulla. 2007. On synopses for distinct-value estimation under multiset

operations. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. 199–210.

[3] Yu Chen and Ke Yi. 2017. Two-level sampling for join size estimation. In *Proceedings of the 2017 ACM International Conference on Management of Data*. 759–774.

[4] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. 2007. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. In *Discrete Mathematics and Theoretical Computer Science*. Discrete Mathematics and Theoretical Computer Science, 137–156.

[5] Yannis E Ioannidis. 1993. Universality of serial histograms. In *VLDB*, Vol. 93. Citeseer, 256–267.

[6] Yesdaulet Izenov, Asoke Datta, Florin Rusu, and Jun Hyung Shin. 2021. COMPASS: Online Sketch-based Query Optimization for In-Memory Databases. In *Proceedings of the 2021 International Conference on Management of Data*. 804–816.

[7] Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter Boncz, and Alfons Kemper. 2018. Learned cardinalities: Estimating correlated joins with deep learning. *arXiv preprint arXiv:1809.00677* (2018).

[8] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. 2015. How good are query optimizers, really? *Proceedings of the VLDB Endowment* 9, 3 (2015), 204–215.

[9] Guido Moerkotte, Thomas Neumann, and Gabriele Steidl. 2009. Preventing bad plans by bounding the impact of cardinality estimation errors. *Proceedings of the VLDB Endowment* 2, 1 (2009), 982–993.

[10] Magnus Müller, Daniel Flachs, and Guido Moerkotte. 2021. Memory-Efficient Key/Foreign-Key Join Size Estimation via Multiplicity and Intersection Size. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 984–995.

[11] Neoklis Polyzotis. 2005. Selectivity-based partitioning: A divide-and-union paradigm for effective query optimization. In *Proceedings of the 14th ACM international conference on Information and knowledge management*. 720–727.

[12] P Griffiths Selinger, Morton M Astrahan, Donald D Chamberlin, Raymond A Lorie, and Thomas G Price. 1989. Access path selection in a relational database management system. In *Readings in Artificial Intelligence and Databases*. Elsevier, 511–522.

[13] David Vengerov, Andre Cavalheiro Menck, Mohamed Zait, and Sunil P Chakkappen. 2015. Join size estimation subject to filter conditions. *Proceedings of the VLDB Endowment* 8, 12 (2015), 1530–1541.