Errata for "Lightning Fast and Space Efficient Inequality Joins" (PVLDB 8(13): 2074-2085)

Zuhair Khayyat[◇] William Lucia[§] Meghna Singh[§] Mourad Ouzzani[§]
Paolo Papotti[†] Jorge-Arnulfo Quiané-Ruiz[§] Nan Tang[§] Panos Kalnis[◇]

[§]Qatar Computing Research Institute, HBKU
[◇]King Abdullah University of Science and Technology (KAUST)
[†]Arizona State University

{zuhair.khayyat,panos.kalnis}@kaust.edu.sa, williamlucia.wl@gmail.com
{mesingh,mouzzani,jquianeruiz,ntang}@hbku.edu.qa ppapotti@asu.edu

This is in response to recent feedback from some readers, which requires some clarifications regarding our IEJoin algorithm published in [1]. The feedback revolves around four points: (1) a typo in our illustrating example of the join process; (2) a naming error for the index used by our algorithm to improve the bit array scan; (3) the sort order used in our algorithms; and (4) a missing explanation on how duplicates are handled by our self join algorithm.

Notice that these four points were addressed in our extended version presented in [2]. We explain each of these points and provide the correction below.

(1) Typo. In Figure 3 of our VLDB paper [1], we wrongly put the value 12 in the second cell of the array L_2. The correct value in the second cell for L_2 should be 9. We corrected this typo error in [2].

(2) Naming error. Although our index to improve the bit array scan follows the same spirit as a Bloom Filter, the index we described in Section 4.1 in [1] is a BitMap index. We corrected this naming error in our journal version [2].

We consider the following query for the remaining two points mentioned above:

```
SELECT r.id, s.id
FROM Employees, Employees s
WHERE r.salary < s.salary AND r.tax > s.tax;
```

(3) Sort order. It is important to clarify that the way we wrote the conditions to perform the sort order (ascending or descending) in Algorithms 1 & 2 was a bit misleading in [1]. We wrongly stated that if the operand in the first non-equi-join condition (*e.g.,* in $r.salary < s.salary$) is $>$ or $\leq$ then the first two arrays on *salary* (*i.e.,* L_1 and L'_1) must be in descending order, otherwise in ascending order. The error is on the kind of operands for L_1 and L'_1 to be on descending order. To be in descending order, the operand must be $>$ or $\geq$. Otherwise (*i.e.,* $<$ or $\leq$), L_1 and L'_1 are in ascending order. The same holds for the array L_2 and L'_2. Our extended paper at VLDBJ [2] has this correction in both Algorithms 1 & 2.

(4) Dealing with duplicates. Overall, the implementation of Algorithm 2 uses a secondary order mechanism to output correct results in the presence of duplicate values. Basically, we first sort on the first attribute we focus on (*e.g.,* salary). In the case of having the same *salary* values for more than one tuple, we sort such duplicates on the second attribute in the non-equi-join condition (*e.g.,* on *tax*). If we still have the same values for *tax*, we finally sort the duplicates based on their tuple ids. These secondary sort orders allow us to produce correct results in the self join case. We did not explicitly state this aspect in Section 3.2 of our original paper [1]. However, we explain so in Section 3.2 of the extended version [2].

1. REFERENCES

[1] Z. Khayyat, W. Lucia, M. Singh, M. Ouzzani, P. Papotti, J. Quiané-Ruiz, N. Tang, and P. Kalnis. Lightning fast and space efficient inequality joins. *PVLDB*, 8(13):2074–2085, 2015.

[2] Z. Khayyat, W. Lucia, M. Singh, M. Ouzzani, P. Papotti, J. Quiané-Ruiz, N. Tang, and P. Kalnis. Fast and scalable inequality joins. *The VLDB Journal*, pages 1–26, 2016.

Proceedings of the VLDB Endowment, Vol. 10, No. 9