Appendix

A1: Scalability of the generated instances of the 10 tables

2o K Products (s) :	
Vendor : 203	Product : 20000
Person : 10210	Review : 200000
ProductFeature : 10519	ProductFeatureProduct : 423222
ProductType : 329	ProductTypeProduct : 80,000
Offer : 400000	Producer : 405
200K Prooducts (m) :	
Vendor : 2027	Product : 200000
Person : 102,596	Review : 2000000
ProductFeature : 47884	ProductFeatureProduct : 3885664
ProductType : 2011	ProductTypeProduct : 1000000
Offer : 400000	Producer : 3956
2000K Products (l) :	
Vendor : 19969	Product : 2000000
Person : 1024622	Review : 19999800
ProductFeature : 94259	ProductFeatureProduct : 38571526
ProductType : 3949	ProductTypeProduct : 10000000
Offer : 40000000	Producer : 39530

A2: Pre-computed inter-source hints

EFJ1 : $\text{Product1}_1 \bowtie_{\text{Product1}_1.nr \equiv_{\mathbb{D}} \text{Product2}_2.nr} \text{Product2}_2 = \emptyset$

EFJ2 : $\text{ProductFeatureProduct1}_1 \bowtie_{\text{ProductFeatureProduct1}_1.product \equiv_{\mathbb{D}} \text{Product2}_2.nr} \text{Product2}_2 = \emptyset$

EFJ3 : $\text{Product1} \bowtie_{\text{Product1}_1.nr \equiv_{\mathbb{D}} \text{ProductFeatureProduct2}_1.product} \text{ProductFeatureProduct2}_1 = \emptyset$

DRS : $\text{ReviewC}_1(...) \equiv_{\mathbb{D}} \text{Review}_5(...)$

$op1(...) \leftarrow \pi_{...}(\text{Offer}_4 \bowtie_{\text{Offer}_4.product=\text{Product1}_1.nr} \text{Product1}_1)$

$op2(...) \leftarrow \pi_{...}(\text{Offer}_4 \bowtie_{\text{Offer}_4.product=\text{Product2}_2.nr} \text{Product2}_2)$

$pfpf1(...) \leftarrow \pi_{...}(\text{ProductFeature}_3 \bowtie_{\text{ProductFeature}_3.nr=\text{ProductFeatureProduct1}_1.productfeature} \text{ProductFeatureProduct1}_1)$

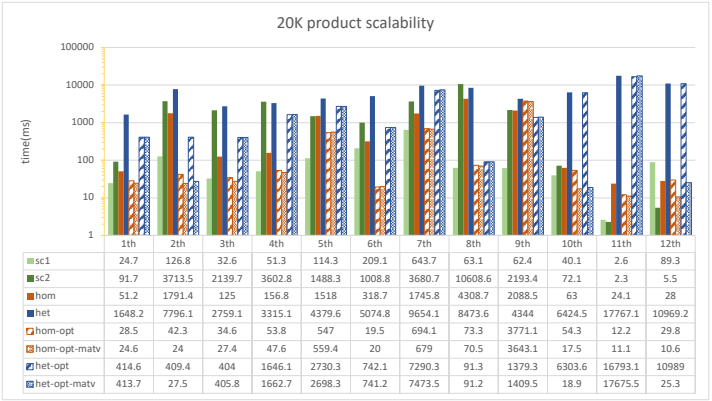
$pfpf2(...) \leftarrow \pi_{...}(\text{ProductFeature}_3 \bowtie_{\text{ProductFeature}_3.nr=\text{ProductFeatureProduct2}_2.productfeature} \text{ProductFeatureProduct2}_2)$

$ppd1(...) \leftarrow \pi_{...}(\text{Product1}_1 \bowtie_{\text{Product1}_1.nr=\text{Producer}_4.product} \text{Producer}_4)$

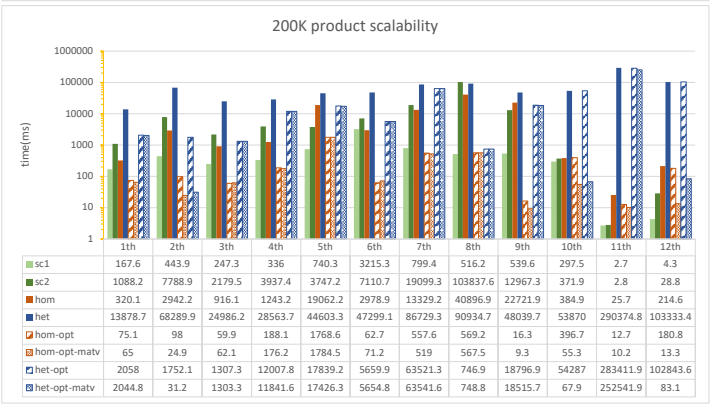
$ppd2(...) \leftarrow \pi_{...}(\text{Product2}_2 \bowtie_{\text{Product2}_2.nr=\text{Producer}_4.product} \text{Producer}_4)$

Here, T_i denotes T is a table/relation in the VDB $\mathbb{\Sigma}$ for data source S_i (or S_i'), and $\mathbb{D}$ is the instance for each instance of the OBDF specification.

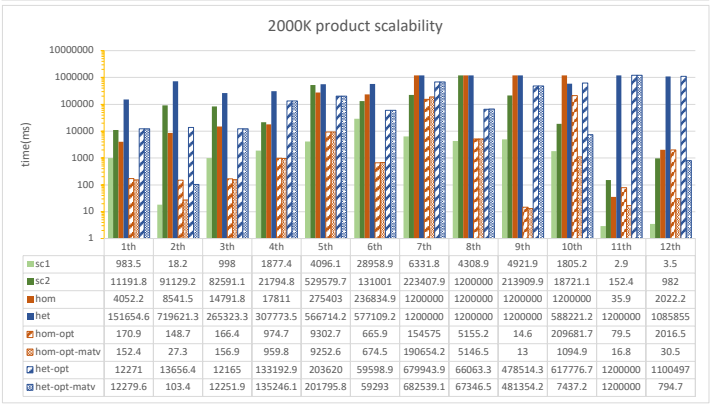
A3: Concrete query evaluation results



	1th	2th	3th	4th	5th	6th	7th	8th	9th	10th	11th	12th
sc1	24.7	126.8	32.6	51.3	114.3	209.1	643.7	63.1	62.4	40.1	2.6	89.3
sc2	91.7	3713.5	2139.7	3602.8	1488.3	1008.8	3680.7	10608.6	2193.4	72.1	2.3	5.5
hom	51.2	1791.4	125	156.8	1518	318.7	1745.8	4308.7	2088.5	63	24.1	28
het	1648.2	7796.1	2759.1	3315.1	4379.6	5074.8	9654.1	8473.6	4344	6424.5	17767.1	10969.2
hom-opt	28.5	42.3	34.6	53.8	547	19.5	694.1	73.3	3771.1	54.3	12.2	29.8
hom-opt-matv	24.6	24	27.4	47.6	559.4	20	679	70.5	3643.1	17.5	11.1	10.6
het-opt	414.6	409.4	404	1646.1	2730.3	742.1	7290.3	91.3	1379.3	6303.6	16793.1	10989
het-opt-matv	413.7	27.5	405.8	1662.7	2698.3	741.2	7473.5	91.2	1409.5	18.9	17675.5	25.3



	1th	2th	3th	4th	5th	6th	7th	8th	9th	10th	11th	12th
sc1	167.6	443.9	247.3	336	740.3	3215.3	799.4	516.2	539.6	297.5	2.7	4.3
sc2	1088.2	7788.9	2179.5	3937.4	3747.2	7110.7	19099.3	103837.6	12967.3	371.9	2.8	28.8
hom	320.1	2942.2	916.1	1243.2	19062.2	2978.9	13329.2	40896.9	22721.9	384.9	25.7	214.6
het	13878.7	68289.9	24986.2	28563.7	44603.3	47299.1	86729.3	90934.7	48039.7	53870	290374.8	103333.4
hom-opt	75.1	98	59.9	188.1	1768.6	62.7	557.6	569.2	16.3	396.7	12.7	180.8
hom-opt-matv	65	24.9	62.1	176.2	1784.5	71.2	519	567.5	9.3	55.3	10.2	13.3
het-opt	2058	1752.1	1307.3	12007.8	17839.2	5659.9	63521.3	746.9	18796.9	54287	283411.9	102843.6
het-opt-matv	2044.8	31.2	1303.3	11841.6	17426.3	5654.8	63541.6	748.8	18515.7	67.9	252541.9	83.1



	1th	2th	3th	4th	5th	6th	7th	8th	9th	10th	11th	12th
sc1	983.5	18.2	998	1877.4	4096.1	28958.9	6331.8	4308.9	4921.9	1805.2	2.9	3.5
sc2	11191.8	91129.2	82591.1	21794.8	529579.7	131001	223407.9	1200000	213909.9	18721.1	152.4	982
hom	4052.2	8541.5	14791.8	17811	275403	236834.9	1200000	1200000	1200000	1200000	35.9	2022.2
het	151654.6	719621.3	265323.3	307773.5	566714.2	577109.2	1200000	1200000	1200000	588221.2	1200000	1085855
hom-opt	170.9	148.7	166.4	974.7	9302.7	665.9	154575	5155.2	14.6	209681.7	79.5	2016.5
hom-opt-matv	152.4	27.3	156.9	959.8	9252.6	674.5	190654.2	5146.5	13	1094.9	16.8	30.5
het-opt	12271	13656.4	12165	133192.9	203620	59598.9	679943.9	66063.3	478514.3	617776.7	1200000	1100497
het-opt-matv	12279.6	103.4	12251.9	135246.1	201795.8	59293	682539.1	67346.5	481354.2	7437.2	1200000	794.7

Fig. 5. Milliseconds used of evaluating the SQL queries over different configurations, where 120000000 (ms) denotes timeout (i.e., the query cannot be evaluated in 20min).

A4: Analysis of the hints for SQL query optimization/simplification

Here, we analyze the performance improved by the inter-source hints in the OBDF settings. For easy describing, u_j, v_k and w_l are used to denote the expressions like π_{a}, σ_{b} and $\pi_{\mathsf{a}}\sigma_{\mathsf{b}}$.

Query q_1-q_6 benefit from the hints of pre-computed empty federated joins, i.e., EFJ1–EFJ3. Such rewriting reduces the computation of federated joins in the SQL queries significantly, such as rewriting the federated joins like:

$$(\mathsf{u}_1(\mathsf{Product1}_1) \cup \mathsf{v}_1(\mathsf{Product2}_2)) \bowtie_\theta (\mathsf{u}_2(\mathsf{Product1}_1) \cup \mathsf{v}_2(\mathsf{Product2}_2))$$

into $\mathsf{u}(\mathsf{Product1}_1) \cup \mathsf{v}(\mathsf{Product2}_2)$, and delivers more computation to local sources to be evaluated in a parallel way. For example, after rewriting terminal federated joins in the query using the approach described from line 11–17 of algorithm $\mathsf{unfold}_{\mathsf{OBDF}}$, the SQL translations q_3 and q_4 of Q_3 and Q_4 are rewritten into the queries with the format $sql_1 \cup sql_2$, where sql_1 is a query only referring source S_1 and sql_2 only referring source S_2 (S'_2), and the SQL translation q_2 of Q_2 is rewritten into q'_2:

$$\begin{array}{l}(\mathsf{u}_1(\mathsf{ProductFeature}_3) \bowtie_{\theta_1} \mathsf{u}_2(\mathsf{ProductFeatureProduct}_1) \bowtie_{\theta_2} \mathsf{u}_3(\mathsf{Product1}_1) \bowtie_{\theta_3} \mathsf{u}_4(\mathsf{Producer}_4)) \cup \\ (\mathsf{v}_1(\mathsf{ProductFeature}_3) \bowtie_{\varphi_1} \mathsf{v}_2(\mathsf{ProductFeatureProduct2}_2) \bowtie_{\varphi_2} \mathsf{v}_3(\mathsf{Product2}_2) \bowtie_{\varphi_3} \mathsf{v}_4(\mathsf{Producer}_4))\end{array}$$

Take the 200K product scalability and hom configuration as an example. Empty federated join based rewriting makes the evaluation times of q_2, q_3 and q_4 reduced respectively from 2942.2 ms, 916.1 ms and 1243.2 ms to 98 ms, 59.9 ms and 188.1 ms (even more faster than in the sc1 situation).

The SQL translations q_6–q_9 of Q_6–Q_9 benefit from redundancy based rewriting. Such rewriting removes the equivalent/containment redundancy w.r.t. the pre-computed data redundancy in the union expressions, such as removing one element from the union $(\mathsf{v}(\mathsf{ReviewC}_1) \bowtie_\theta \mathsf{A}) \cup (\mathsf{u}(\mathsf{Review}_5) \bowtie_\theta \mathsf{A})$ by taking data distribution and heterogeneity into consideration, so as to reduce the computation of federated joins, such as rewriting the join:

$$(\mathsf{u}_1(\mathsf{ReviewC}_1) \cup \mathsf{v}_1(\mathsf{Review}_5)) \bowtie_\theta (\mathsf{u}_2(\mathsf{ReviewC}_1) \cup \mathsf{v}_2(\mathsf{Review}_5))$$

as $\mathsf{u}(\mathsf{ReviewC}_1)$ or $\mathsf{v}(\mathsf{Review}_5)$, and the access of inefficient data sources. For example, after redundancy based rewriting, q_9 is rewritten as a query only referring source S_2 (S'_2). After removing redundancy, the evaluation of these three queries is as fast as in the sc1 situation.

The SQL query translation q_2, q_{10} and q_{12} of the SPARQL queries Q_2, Q_{10} and Q_{12} benefit from the materialized views. Here q'_{10} and q_{10} are the same, and q'_{12} and q_{12} are the same. For example, after applying all the hints, q_2 is further rewritten as:

$$q''_2 : (\mathsf{u}(S^{\mathsf{M}}.ppd1) \bowtie_\theta \mathsf{v}(S^{\mathsf{M}}.pfpf1)) \cup (\mathsf{u}'(S^{\mathsf{M}}.ppd2) \bowtie_\varphi \mathsf{v}'(S^{\mathsf{M}}.pfpf2))$$

which is a query over S^{M}. No computation of federated joins and accessing of inefficient sources is needed. In the 2M product scalability, for hom and het, it makes the query evaluation time further reduced respectively from 8541.5 ms and 719621.3 ms to 27.3 ms and 103.4 ms. Moreover, q_{10} is the query:

$$\begin{array}{c}((\mathsf{u}_1(\mathsf{Product1}_1) \bowtie_{\theta_1} \mathsf{v}_1(\mathsf{Offer}_4) \bowtie_{\varphi_1} \mathsf{w}_1(\mathsf{Vendor}_4))) \cup \\ ((\mathsf{u}_2(\mathsf{Product2}_2) \bowtie_{\theta_2} \mathsf{v}_2(\mathsf{Offer}_4) \bowtie_{\varphi_2} \mathsf{w}_2(\mathsf{Vendor}_4)))\end{array}$$

Based on the materialization *op1* and *op2* in S^{M}, q_{10} is finally rewritten as the query q''_{10} : $(\mathsf{u}(S^{\mathsf{M}}.op1) \bowtie_{\theta'} \mathsf{v}(\mathsf{Vendor}_4)) \cup (\mathsf{u}'(S^{\mathsf{M}}.op2) \bowtie_{\varphi'} \mathsf{v}'(\mathsf{Vendor}_4))$. Again take 2M products scalability as an example. In the federation setting, i.e., hom and het, the rewriting reduced the evaluation times respectively from 209681.7 ms and 617776.8 ms to 1094.9 ms and 7437.2 ms. Although in the heterogeneous situation, S'_4 is a set of CSV files, it does not effect the overall performance much, since the source only needs to evaluate simple selecting queries.

In summary, the overall evaluation results indicate the rationality and importance of our research as well as the high-efficiency and power of the proposed solution in improving the overall all query answering performance.