Locality-Aware Join Planning with Reserved Capacity

Amalia R. Nzegho
Laboratory for Data Systems, Institut Méridienne
Lausanne, Switzerland
amalia.nzegho@meridienne.example

Henrik V. Rautiala
School of Computing, Aurajoki Institute of Technology
Turku, Finland

Elif S. Orbayhan
Department of Information Engineering, Ekinova University
İzmir, Türkiye

Lucien K. Barrot
Laboratory for Data Systems, Institut Méridienne
Lausanne, Switzerland

capstract:2611.00003v1 [cs.DB] 2 Nov 2026

Abstract

Physical query plans normally take the locations of input records as given. In a class of intermittently connected deployments, record placement can still be changed when the query is known, subject to a hard limit on occupancy and a cost for changing existing assignments. We study this problem at seated receptions, where participants carry input records and each dining table is a local execution site. A cost model jointly selects a seating plan and a physical join plan. It distinguishes positive rows from completeness evidence so that an empty chair can provide evidence for a left outer join whose preserved row is located elsewhere. On 120 held-out seating instances drawn at three scales, all evaluated plans agree with centralized relational evaluation. At 96 guests the joint optimizer reduces the median number of boundary-crossing records from 146 to 53 for a grouped program query and from 215 to 76 for a sponsor lookup. Central collection is cheaper for the triangle workload. When absent-carrier acknowledgments are dropped from the protocol, 8.8% of sponsor answers at 96 guests with a vacant reservation report absence before the inner relation is closed. We quantify the cost of retaining a place for a carrier that never arrives.

CCS Concepts

• Information systems → Query optimization; • Information systems → Distributed database management systems.

Keywords

query optimization, join locality, outer joins, distributed query processing, completeness

Reference Format:
Amalia R. Nzegho, Henrik V. Rautiala, Elif S. Orbayhan, and Lucien K. Barrot. 2026. Locality-Aware Join Planning with Reserved Capacity. Preprint, November 2, 2026.

1 Introduction

In a seating plan the organizer assigns each confirmed guest a chair. Movement thereafter is possible, but individual conversations and the exchange of small objects often take place within the assigned table. Printed programs and cards can carry information about the guest to whom they were issued. To answer a question about attendees, the organizer generally walks to several tables or collects the relevant cards first.

We consider the case in which the question is known when the seating chart is prepared. Two people who would otherwise have to pass a card between tables can sometimes be seated at the same table. The last change to the chart is therefore a data placement operation. A venue may specify both a seating plan and a set of scheduled collection rounds, making it possible to optimize placement along with the order of relational operators. The two schedules have distinct deadlines. Entries can be copied onto cards during service. After service begins the guests representing those entries cannot be repartitioned without charging for a seat change. A seating change has unit cost per guest even if that guest carries multiple rows.

Consider a guest whose party sponsor is expected at a different table. A table representative can look up a sponsor and find nobody sitting in the expected seat. The sponsor relation remains open until the attendance window closes. If it turns out that the sponsor never arrived, the guest still appears in the answer, with a null sponsor. The room therefore needs a collection stop at an unoccupied chair. A courier records absence there once the arrival window closes.

2 Background

System R jointly chose access paths and join orders given a cost model for prematerialized relations [16]. General accounts distinguish plan enumeration from statistics and cost estimation [4]. Graefe [7] describes the role of physical properties such as sort order. Within a reception, co-location at a specified round is a physical property of a stream of records. A rearrangement of chairs can establish that property for several streams at once, subject to reserved capacity. The arrangement also establishes a practical limit on which stream properties can coexist. A guest who carries several logically unrelated rows has only one seat.

Distributed semijoins reduce data transmission by exchanging keys before rows [2]. Acyclic joins permit two passes of semijoin reduction [19]. Multiway joins can avoid large binary intermediates on cyclic patterns [1, 14]. Our central collector can use either approach. The controlled distributed plans use physical merge and lookup operators, with dictionary projection where the same keys recur. For small queries the overhead of getting the first key list across the room can dominate the savings in projected rows. Heikkuri and Otero Valls [8] measured this overhead when evaluating selection on paper ballots passed down rows of seats. Mirecourt and Afenyo [12] studied manually maintained room directories with hard capacity constraints. That work assumes that a row exists at every designated site.

Codd's relational model separates named data from its physical location [5]. The uncertain presence of a row introduces a

separate question about whether a site has delivered all rows for a key. The distinction between known data and evidence of completeness has been studied for data integration [11, 13]. Calarco and Peduru [3] describe event rosters that close locally before the venue's aggregate count closes. Their totals include an explicit count of guests whose seats were never occupied. We use a sealed roster at each site to supply key-level closure metadata, including keys whose only possible carrier did not take a seat. An inactive site still requires a source for its seal, as observed in Kandemir and Stornes [9]. Marking it inactive in an allocation plan does not generate an execution event.

3 Representation and execution

3.1 Sites, carriers, and records

Let G be the announced guest roster and T a fixed set of tables. Table t has b_t seats. Each g in G has a reference table r_g, given by the initial alphabetical seating plan, and receives one or more printed records before service. Each record belongs to one relation and has a row identifier. Multiple records can agree on all query-visible attributes. We keep their identifiers through exchange for bag semantics and discard them only at the last projection or aggregation. An arrival flag selects all records on a guest's card together. A printed roster lists the possible carriers of an inner key. A roster entry alone supplies no row to the query.

A layout ℓ assigns each roster member to one table. Some groups must stay together in the input. The generator gives each two-person required group a single placement variable with weight two. Those constraints model seating requests recorded before the query was registered. The optimizer cannot force an arrival. Capacity is charged to assigned seats, which include places reserved for people who may be late or absent. The organizer can release an unused reservation after the arrival cutoff. It cannot be released in time to justify a seating plan that overflows when every reserved guest arrives.

Each table has a representative with a local ledger. The representative sees arrivals at the table and can copy rows from present cards. Between tables the only transport is a courier following a published round schedule. Courier stops visit all tables in ascending order, then the collector. An envelope can contain at most eight data records. To remove leverage from lucky adjacency within a round, all envelopes are addressed just before a round and all deliveries are visible to their recipients at its end. A sender can make local copies without incurring network traffic. Sending the same projected witness to two sites costs two records. Control receipts carry round and batch identifiers and have fixed-size headers. We count all row, key and seal payload records in the main endpoint and record envelope counts separately.

3.2 Completeness and null extension

For an outer join $\mathrm{leftjoin}(P, S)$ on key k, define $C(k)$ to be the sites to which any authorized carrier of an S row with key k is assigned. The coordinator roster identifies $C(k)$ after layout selection, including reservations for absent carriers. A site issues a seal for k when its local attendance window and outgoing S batch for k close. The ledger runner issues a zero seal for an unattended table after checking its attendance sheet at cutoff. A seal contains the number of rows in the batch and the final batch identifier. Each row and each seal names its source. The receiving site counts distinct row identifiers per batch. A null-extended output for a preserved row p is legal at receiver u when

$$\begin{aligned} &\forall t \in C(p.k), \quad \mathrm{sealed}_u(t, p.k) \\ &\qquad \wedge\, \mathrm{received}_u(t, p.k) = \mathrm{declared}_u(t, p.k), \\ &\sum_{t \in C(p.k)} \mathrm{received}_u(t, p.k) = 0. \end{aligned} \tag{1}$$

When $C(p.k)$ is empty, the roster has already certified the key outside the inner relation's domain. We emit the null extension locally in that case. Otherwise the receiving site waits according to Eq. (1). Positive matches can stream while it waits. A retransmission with the same row identifier is acknowledgment retry at the physical layer and never counts twice. Appendix A gives the argument for bag equivalence.

4 Optimization

4.1 Cost

The planner takes estimated key frequencies and per-carrier arrival probabilities from the roster. Within a given layout it enumerates three physical plan families for each query, local-first with evidence exchange, key request followed by filtered row exchange, and direct collection. Program aggregation may also ship a partial count rather than each matched row. For triangle evaluation we allow central multiway matching and distributed binary plans with three choices of first edge. We require that each required result and each relevant seal have a route to the designated consumer. Infeasible candidates receive infinite cost.

For π a physical plan, $W(\ell, \pi)$ is its predicted count of crossing records, $D(\pi)$ the number of exchange rounds on its critical path, and $M(\ell)$ the number of guests assigned away from the reference table. The objective is

$$J(\ell, \pi) = W(\ell, \pi) + 8D(\pi) + \lambda M(\ell), \qquad M(\ell) = \sum_{g \in G} \mathbf{1}\{\ell(g) \neq r_g\}. \tag{2}$$

We price each round at the payload capacity of one envelope and use a seat-change price of half a record. The prediction for a seal record is one when the plan ships evidence for a requested key whose candidate roster at a source is nonempty. Its contribution is independent of the attendance of the candidates. Row predictions multiply estimated co-key multiplicities by arrival probabilities, with one exception for rows on the same carrier. Those rows are either all present or all absent; their match probability uses a single arrival factor. The simulator uses a common random stream for the attendance realization of a roster across all physical plans. Local statistics see assigned rows, while the benchmark evaluator sees realized rows. We use a sound route structure for completeness independently of any cost approximation.

4.2 Search

Our optimizer starts from the reference seating chart and costs every physical alternative. At each iteration it visits legal pairs of guests and evaluates a swap, including swaps with an unused unreserved position. A swap between two required groups is legal only if the new sites still meet constraints. It visits all feasible single relocations too. It then applies the best improving layout move with the best plan under that layout. We stop when no move improves Eq. (2) or after 30 improving moves. A lexicographic tiebreak, based on roster ids and plan family, makes repeated runs deterministic. The procedure uses the same physical candidates as the fixed placement comparator. It can skip rescoring a key when none of that key's producer or consumer assignments changed. That optimization follows the delta scoring approach of Versnel and Oum [18] for seating repair.

A second placement baseline chooses swaps based on the number of related guest pairs made local. It finishes allocating chairs before asking the same physical enumerator to choose a join plan. It receives no reward for a null match and treats a many-to-one grouping as a sum of pair affinities. Closely related placement heuristics occur in hypergraph partitioning [10]. Our baseline leaves the eight-record envelope limit and the seat-change penalty to the physical cost calculation after seating. Their omission from placement distinguishes a workload affinity signal from the end-to-end objective.

For a check on the optimizer, 24 additional rosters have three four-seat tables and at most nine assigned places. We enumerate all group-feasible layouts and physical plans for these small instances. The comparison is with the same objective, probabilities and plan families, giving a lower bound on the achievable cost within the search domain. It does not certify that the candidate set includes the least expensive conceivable plan. We hold the invitation list fixed even in the exhaustive search.

5 Evaluation

5.1 Workloads and design

We generate 150 rosters, balanced across 48, 96 and 192 assigned guests, using ten rosters per size for tuning and 40 for testing. A table has 12 numbered places. The three sizes have 5, 9 and 18 tables, respectively. Unless stated otherwise, every announced carrier is allocated a seat, and disposable seats that start empty may be used by the optimizer. The carrier arrival probabilities range from 0.58 to 0.98. Within two-person groups an unobserved common shock may prevent both from arriving. The optimizer sees marginal probabilities, while the scorer draws the joint outcome. All 120 test rosters carry the inputs for all three queries, and each query uses the same attendance realization on a given roster. We report one measured execution per query and plan on each roster. The 24 smaller rosters used for enumeration have separate identifiers.

Each card associates a carrier with a party code, and some cards carry sponsor rows for party codes. Other cards list printed introduction permissions in three edge roles. Values are synthetic integer identifiers. Roster order and party grouping are sampled independently before enforcing the placement requirements. Party sizes follow a truncated discrete distribution with median four and maximum twelve. We draw duplicate sponsor row values, with distinct row identifiers, on 10% of the sponsor carriers. Hosts are ordinary record carriers with assigned seats. A host can be a sponsor for more than one party. Missing sponsor rows represent absent carriers or codes without a declared sponsor, depending on the generator flag. Table 1 gives the three tasks.

Table 1: Workload definitions. Each query is evaluated separately and has a designated result table. The host dimension in Program is keyed by party code.

Workload	Operators	Output
Program	Carrier-to-host inner lookup, local program lookup, grouped count	Count per program class and host region
Sponsor	Guest left outer join with all realized party sponsor rows	Guest, sponsor identifier or null
Introduction	Cyclic three-way join on directed guest pairs	Ordered triples of mutually eligible guests

For Program, the program value is local to the guest and the host dimension is unique per party code. Only guests with a realized row in both dimensions contribute to a count. The carrier bearing the host dimension can be absent independently of the others in the party. For Sponsor, the preserved relation contains one row per arrived guest and the inner relation can contain multiple rows for one code. The output has one tuple per match, or one null tuple if none exists. For Introduction, three separately generated edge lists point from a to b, b to c and c to a, with records assigned to the first endpoint. Self edges are prohibited by the generator. Positive edge records travel only if their carriers arrived. A missing assignment to a relation is empty for the duration of the episode.

We run a separate stress test of absence evidence. In the core suite the attendance cutoff precedes dispatch of any completeness seal. The selected guest-to-seat assignment has no effect on the actual attendance draw. In the stress run each sponsor reservation is resolved at one of the first four dispatch rounds, drawn uniformly. In that run the oracle uses the final snapshot. Every possible carrier has a final status in the snapshot. The local ledger learns that status when the carrier arrives or its reservation expires. We compare protocols that wait for this evidence with a premature protocol that treats an empty reservation as final on its first visit.

5.2 Baselines and measurements

Fixed seating represents the unmodified chart, using the same physical plan families and Eq. (2). To maintain a genuine distributed comparator, fixed and affinity rows in the main results report the cheapest candidate among the two distributed families.

The joint optimizer row also reports the best distributed candidate, so differences measure layout and distributed-plan effects. Central collection transfers every input row to the designated result table if its carrier is elsewhere, and computes the whole query there. It may issue a roster digest when a relation has no rows. It does not have to route a seal for every key. It already owns the combined roster at the end of the collection window. We include its median alongside the distributed plans. In the unrestricted choice the optimizer can select it.

A centralized exact evaluator uses the run’s realized occurrence ids and applies bag semantics, SQL null extension and exact integer aggregation. We sort the outputs on all result fields and compare multiplicities, treating null as a distinct comparison symbol in the test harness. With a zero result set we require a valid delivery trace even if the equality test is trivial. We never exclude a run because a result was empty. Six rosters at each scale were replayed with printed cards, assigned labels on chairs and envelopes on a tabletop floor map. This replay moves records and paper labels rather than people, with arrival flags revealed on the same round schedule. An independent tally counts transferred records, envelope occupancy and final multiplicity. This check tests bookkeeping under a finite exchange schedule. Its execution time does not estimate reception duration.

The primary contrast is the paired difference in crossing records for Program at 96 guests, fixed minus joint. We summarize contrast and uncertainty with the median paired difference and a 10,000-draw percentile bootstrap over independently generated rosters [6]. Per-workload medians are computed over the 40 rosters at the relevant scale. Marginal medians may differ from the median of paired differences. Other workload comparisons and stress findings are reported descriptively. Eight tuning rosters per size were enough to choose the move limit on the development curves. The remaining two per size checked roster feasibility and batching before we collected the test outcomes.

6 Results

6.1 Work by placement and query

Table 2 compares the four plans. For Sponsor, the joint optimizer spends some seat changes on colocating the senders and consumers of seals for sponsors with low estimated attendance.

Table 2: Median records crossing a table boundary at 96 assigned guests (40 rosters). Counts include data and completeness payloads. Lower is better.

Placement and execution	Program	Sponsor	Introduction
Fixed seating, distributed	146	215	306
Affinity seating, distributed	81	132	285
Joint seating and plan	**53**	**76**	246
Fixed seating, collect	116	137	**203**

For Program the median paired saving is 89 records (95% interval 80 to 97), compared with a difference of 93 between the two marginal medians. The joint plan typically sends partial counts toward the end of the dispatch schedule. Figure 1 gives the scale dependence and the median seat change cost on the same rosters. At the largest size the assigned guest count doubles and the relative saving is similar. Changes to the grouping distribution could shift this relationship, since plan benefits depend on the share of keys that fit at one or two tables. At 96 guests the 22 moved guests in the median instance occupy fewer than two full tables and are drawn from six tables in the median move trace.

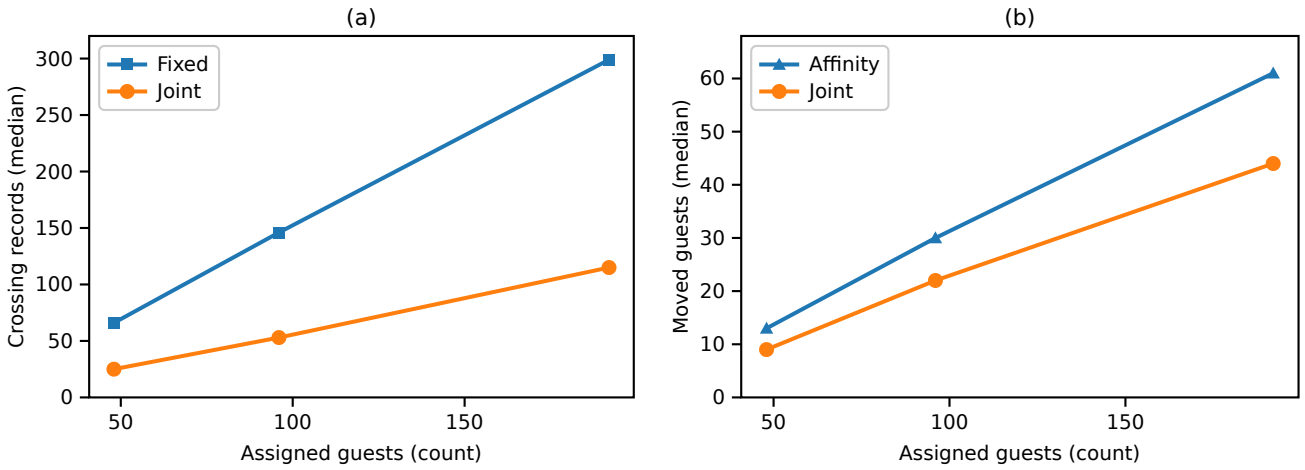


Figure 1: Program workload at three scales. Left shows median boundary-crossing records. Right shows median guests seated away from their reference table.

Introduction produces less shareable local work. Each edge’s key is an endpoint pair, and interest in the same pair is spread across enough locations that shifting a carrier into one favorable table often moves it out of another. Central collection wins the median comparison and is selected in 31 of 40 Introduction rosters at the medium size when all alternatives are eligible. At 96 guests the median envelope counts for joint Program, Sponsor and Introduction are 18, 25 and 39. The median round counts are

two, three and three. Although the record endpoint ignores envelope headers, envelope counts preserve the median Program and Sponsor rankings against fixed placement (31 and 42 envelopes). The direct collector uses a single exchange round after the cutoff for all three queries.

The optimizer reached a local minimum before the move cap on 114 of 120 Program runs. On the 24 smaller rosters it found a minimum-cost Program arrangement in 19, with relative cost gaps of 1.4% to 7.8% on the other five. Local minima are often caused by a nearly full table. For one small case the optimum exchanged three carriers in a cycle. The corresponding pairwise changes would have exceeded capacity or separated a required pair during the intermediate step. Requiring a legal arrangement at every iteration ruled out both pairwise routes. The optimum remained a legal starting layout if specified all at once.

6.2 Vacant reservations

The four physical plans in the main comparison agree with centralized evaluation on all 360 roster-query instances. We saw no lost or duplicate output in the 18 paper replays or a mismatch between the simulator's payload count and the envelope tally. The premature protocol fails under staggered arrival. Among the 40 medium-size Sponsor rosters, there are 398 preserved rows present at the first visit whose possible inner carriers include at least one vacant reservation. In 35 of those cases the premature evaluator emits null for a preserved row that later receives an actual sponsor match (8.8%). The other 363 premature checks happen to be correct for that row, or wait because the executor already sees at least one positive match. We report errors per preserved row. A single early seal can affect several rows on two tables, so an interval treating 398 errors as independent Bernoulli trials would be inappropriate.

Table 3: Sponsor stress run at 96 assigned guests. Affected rows have at least one possible inner carrier in a reserved seat vacant at an early dispatch.

Evidence rule	Affected rows	Incorrect nulls
Close at first empty visit	398	35
Close after complete seal	398	0

Figure 2 shows what the flawed executor knows at each dispatch. Out of the 398 rows, 35 are initially eligible for its incorrect null extension and receive a match later. The number of those false nulls that would still be present in the answer after the next delivery is 28 if revisions are permitted. In the implemented flawed protocol a null already delivered cannot be retracted. The plotted counterfactual therefore isolates the duration of the information gap and is distinct from errors in delivered results. Calarco and Peduru [3] discuss the problem of sending corrected headcounts after service orders have been handed to a different room.

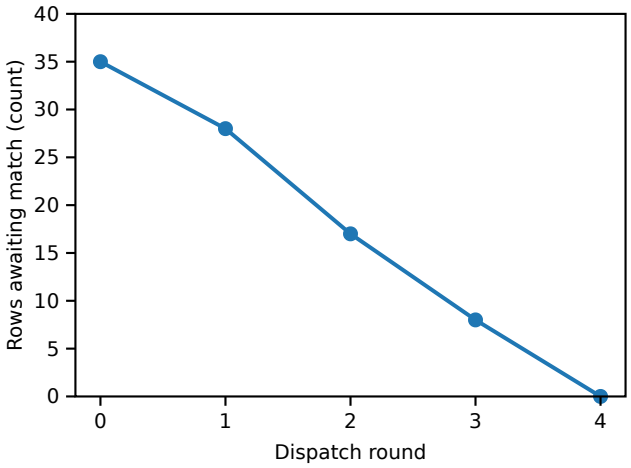


Figure 2: Sponsor rows in the premature-null cohort still awaiting their first positive match after each dispatch. This cohort is defined from replay against the final snapshot.

In a separate sweep, we suppress the Sponsor inserts on independently sampled carrier cards while preserving the realized guest relation, roster and physical plan. Figure 3 shows realized transfers when the extra probability of an empty Sponsor card is 0%, 25%, 50% or 75%. At the last point the median total falls from 76 to 59. The same seal traffic is required because those keys still have possible source sites. A site whose cards lose all of their sponsor entries can finish the run having supplied only certificates.

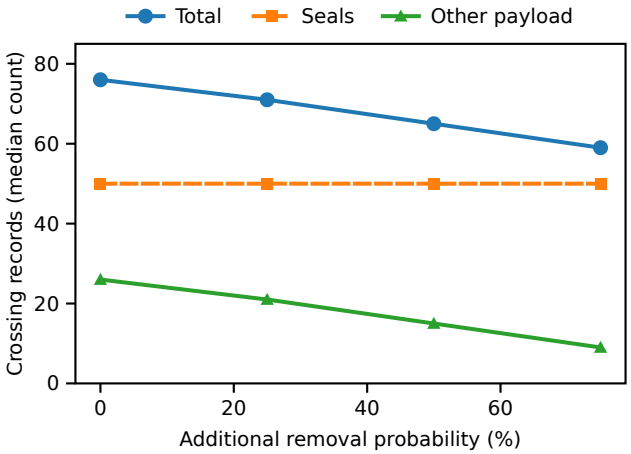


Figure 3: Sponsor transfer under extra thinning at 96 guests. The total is the median of per-roster totals. Component medians are taken separately.

7 Discussion

This evaluation holds the relation between a guest and the records on that guest's card fixed. It is possible to print a second copy of every row before the event. If available snapshot information makes those copies authoritative, the optimizer can sim-

ply ship paper to the collector before moving a guest. In our model any pre-event copy represents an expected row. It becomes valid only when synchronized with the actual arrival record. One could reduce runtime state to those activation records and pre-distribute the data values. This is a different plan with an additional storage and coordination cost. It could be valuable for stable lookup dimensions and races with the onset of service when a late change to the roster is expected [15].

The model assumes a single layout per query. At an actual reception there may be several queries with different consumers and no opportunity to reseat guests between them. A physical layout picked for sponsor completeness may put the wrong groups together for introduction. Joint placement for multiple queries would require weighing their predicted join costs on one common assignment, as in multi-query optimization [17]. The benefit of reusing a seal would then depend on whether both joins interpret its inner snapshot at the same cutoff. A query issued at the end of the first course could be answered correctly and still fail verification when evaluated by the roster from the second course. For a single query the place can be released as soon as the ledger runner records its absence. The seal leaves on the next dispatch even if the chair has already been carried away.

A Correctness of outer evaluation

Assume the registry of possible inner carriers is sound and complete, each carrier has exactly one assigned site and the seal protocol captures all rows in the final snapshot. Then any matching inner row for preserved row p comes from a site in $C(p.k)$. Each delivered match with a distinct occurrence id makes one output. Joining a preserved duplicate independently is necessary to implement bag semantics. Once all declared counts agree with the received counts, no missing row can change the observed multiplicity. If the sum in Eq. (1) is zero, no match can ever be delivered for that key under the assumed snapshot, and a single null row is required. Conversely, when the sum is positive the executor can produce only the corresponding matches. The registry-empty case follows because no permissible source can supply that key.

B Cost replay and reproducibility

On a single 3.4 GHz x86-64 core, the median full optimization time at 96 assigned guests was 210 ms for Program, including catalog construction and all candidate evaluations. The 95th percentile was 430 ms. Execution simulation and centralized replay took a further 14 ms at the median. The printed replay counts crossings at envelope delivery, with a separate mark per copied row when a payload is replicated. Its verification record contains no arrival identifiers beyond the synthetic roster. The implementation does not infer unopened envelope contents from the name written on the chair.

Acknowledgments

We thank Sibel N. Akveren for checking delivered occurrence counts and Janne Ilvesniemi for recovering the unused place labels after the third replay.

References

[1] Albert Atserias, Martin Grohe, and Dániel Marx. 2013. Size Bounds and Query Plans for Relational Joins. *SIAM Journal on Computing* 42 , 4 (2013), 1737–1767.

[2] Philip A. Bernstein and Dah-Ming W. Chiu. 1981. Using Semi-Joins to Solve Relational Queries. *Journal of the ACM* 28 , 1 (1981), 25–40.

[3] Daria L. Calarco and Nalin K. Peduru. 2023. Arrival Closure in Distributed Event Rosters. *Journal of Venue Operations* 18 (2023), 56–81.

[4] Surajit Chaudhuri. 1998. An Overview of Query Optimization in Relational Systems. In *ACM Symposium on Principles of Database Systems*. 34–43.

[5] Edgar F. Codd. 1970. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM* 13 , 6 (1970), 377–387.

[6] Bradley Efron and Robert J. Tibshirani. 1993. *An Introduction to the Bootstrap*. Chapman and Hall.

[7] Goetz Graefe. 1993. Query Evaluation Techniques for Large Databases. *ACM Computing Surveys* 25 , 2 (1993), 73–169.

[8] Aino M. Heikkuri and Manuel Otero Valls. 2022. The Cost of Passes in Paper-Based Selection. In *Workshop on Intermittent Data Processing*. 9–17.

[9] Nehir L. Kandemir and Joakim P. Stornes. 2021. Evaluating Lookups at Reserved but Inactive Sites. In *Workshop on Intermittent Data Processing*. 44–52.

[10] George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. 1999. Multilevel Hypergraph Partitioning: Applications in VLSI Domain. *IEEE Transactions on Very Large Scale Integration Systems* 7 , 1 (1999), 69–79.

[11] Alon Y. Levy. 1996. Obtaining Complete Answers from Incomplete Databases. In *International Conference on Very Large Data Bases*. 402–412.

[12] Noémie J. Mirecourt and Kweku N. Afenyo. 2024. Locality and Fixed Capacity in Manually Maintained Directories. *Journal of Practical Data Management* 14 , 2 (2024), 131–158.

[13] Amihai Motro. 1989. Integrity = Validity + Completeness. *ACM Transactions on Database Systems* 14 , 4 (1989), 480–502.

[14] Hung Q. Ngo, Ely Porat, Christopher Ré, and Atri Rudra. 2012. Worst-Case Optimal Join Algorithms. In *ACM Symposium on Principles of Database Systems*. 37–48.

[15] Maria C. Savio and Eelco van der Klei. 2022. Snapshot Validation for Advance Printing of Mutable Event Records. In *Workshop on Data Management at the Edge*. 21–29.

[16] Patricia G. Selinger, Morton M. Astrahan, Donald D. Chamberlin, Raymond A. Lorie, and Thomas G. Price. 1979. Access Path Selection in a Relational Database Management System. In *ACM SIGMOD International Conference on Management of Data*. 23–34.

[17] Timos K. Sellis. 1988. Multiple-Query Optimization. *ACM Transactions on Database Systems* 13 , 1 (1988), 23–52.

[18] Jacqueline M. Versnel and Davin Oum. 2020. Incremental Scoring for Late Repair of Printed Place Assignments. *Journal of Venue Operations* 15 (2020), 109–129.

[19] Mihalis Yannakakis. 1981. Algorithms for Acyclic Database Schemes. In *International Conference on Very Large Data Bases*. 82–94.