RELCRAFT: CREDIT-GUIDED AND POINT-IN-TIME-SAFE PROGRAM SEARCH FOR RELATIONAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) agents can automate relational feature engineering by searching over executable SQL. However, existing agents face three failure modes: free-form generation often produces invalid or temporally unsafe programs, scalar validation feedback poorly identifies which feature block helped, and repeated reuse of one validation set leads to optimistic selection. We introduce RelCraft, a reliability-oriented framework with one component for each failure mode. A schema-typed grammar and a deterministic point-in-time verifier restrict search to executable, row-aligned programs. Paired cross-fitted coalition evaluation estimates each block's average marginal utility under matched model fits. A protected split then selects a frozen SQL–predictor pair using a fixed complexity prior, without returning selection outcomes to the LLM. We give a finite-sample guarantee for the bounded group loss used by this selector. On 32 tasks from three relational benchmark suites, RelCraft is best or second best on all five suite-level metrics; relative to RelAgent, it lowers the SQL failure rate from 6.8% to 1.9%, reduces the protected–test gap from 1.31 to 0.56 points, and selects 32% fewer feature blocks, at the cost of 52% more offline cross-fit jobs.

1 INTRODUCTION

Relational prediction (Aditya et al., 2002) still requires task-specific choices: which key paths to traverse, which events precede a target cutoff, how far back to look, and how rows should be aggregated (Fey et al., 2024). The number of plausible programs grows rapidly with schema connectivity, while a single unsafe join can make an otherwise strong predictor invalid. Relational neural and foundation models reduce manual engineering but fix much of this access pattern in their architectures (Robinson et al., 2024; Chen et al., 2025; Dwivedi et al., 2026; Ranjan et al., 2026; Hudovernik et al., 2026). An alternative is to let an LLM write SQL features, fit a tabular model, and revise the program from validation feedback (Hollmann et al., 2023; Han et al., 2024; Kim et al., 2026; Huang et al., 2026). This separation is attractive: the LLM is used during development, whereas deployment retains deterministic SQL and a conventional predictor. Existing systems, however, still couple semantic planning, executable code, and model selection inside scalar trial-and-error.

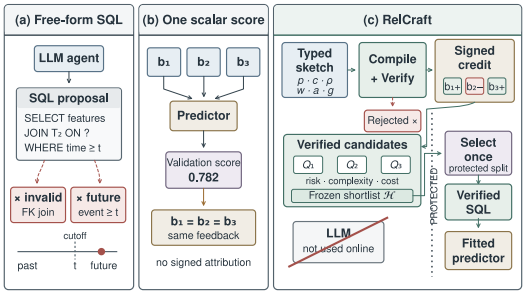

Figure 1: From scalar trial-and-error to reliable relational program search. RelCraft replaces free-form SQL and undifferentiated feedback with typed verification, block credit, and one-way protected selection; only verified SQL and a tabular predictor remain at inference.

This coupling leads to three failure modes. **Weak constraints.** Executable SQL can still join an undeclared key, mix prediction times, use post-cutoff events, or expand target rows (Zhong et al., 2020). Such errors may survive a syntax check and produce deceptively strong validation scores; prompt reminders do not establish point-in-time safety (Kim et al., 2025). **Ambiguous feedback.**

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

When one trial changes several feature blocks and the predictor, a scalar AUROC or MAE cannot identify which edit helped. Conversely, two jointly useful blocks may each look useless when tested alone (Kim et al., 2024). **Adaptive optimism.** Dozens of trials and multiple rollouts repeatedly optimize against the same development split. Selecting the largest observed score rewards both genuine quality and favorable noise, especially on small temporal tasks.

RelCraft changes the search unit from raw SQL to a verified relational sketch and the feedback unit from a trial score to paired block credit. Its compiler handles syntax, key traversal, grouping, and cutoff predicates; its coalition evaluator compares a block across compatible subprograms on identical folds and seeds; and its protected selector is outside the adaptive LLM loop. Figure 1 summarizes this shift. Our contributions are: (i) a typed, point-in-time-checked program language; (ii) interaction-aware cross-fitted credit with explicit attribution cost; (iii) closed joint feature–predictor search whose final selection has a finite-sample guarantee for a bounded group loss; and (iv) a 32-task evaluation that distinguishes predictive quality, agent decision efficiency, actual model-fitting cost, reliability, and deployment behavior.

2 Related Work

2.1 Relational Representation Learning

Relational GNNs and transformers propagate information over schema-induced graphs (Robinson et al., 2024; Chen et al., 2025; Dwivedi et al., 2026; Wang et al., 2025; Ranjan et al., 2026); RDBLearn and KumoRFM-v2 are pretrained models that make in-context predictions over relational databases (Zhang et al., 2026; Hudovernik et al., 2026). These models learn representations inside a fixed relational computation. RelCraft instead searches an explicit, task-specific SQL program and a tabular predictor.

2.2 LLM Agents for Feature Engineering

LLM feature systems synthesize flat-table transformations (Hollmann et al., 2023; Han et al., 2024; Abhyankar et al., 2026; Nam et al., 2024), while ReFuGe and RelAgent extend agentic search to relational databases (Kim et al., 2026; Huang et al., 2026). RelCraft differs in where control is placed: typed compilation precedes execution, block feedback is estimated through matched coalition comparisons, and a protected split is never exposed to the agent. Deep Feature Synthesis (Kanter & Veeramachaneni, 2015) enumerates aggregation features along key paths without an LLM; RelCraft's typed sketches play a similar role but are proposed by an agent and verified for point-in-time safety. Block credit follows Shapley-style attribution (Shapley, 1953; Lundberg & Lee, 2017), and the protected split is related to holdout reuse in adaptive data analysis (Dwork et al., 2015) and to complexity-regularized selection; here it is implemented as a one-way boundary for database agents.

3 Problem Setup

Let $D = \{T_1, \ldots, T_K\}$ be a relational database whose tables are linked by a primary–foreign-key schema graph G_D. A target is $z_i = (e_i, t_i, y_i)$, where e_i is an entity, t_i is its prediction cutoff, and y_i is its label. Chronological targets are divided into training, adaptive development, protected selection, and test sets, where $\prec$ means that every cutoff in the left set precedes every cutoff in the right set:

$$\mathcal{T}_{\mathrm{tr}} \prec \mathcal{T}_{\mathrm{dev}} \prec \mathcal{T}_{\mathrm{sel}} \prec \mathcal{T}_{\mathrm{te}}. \tag{1}$$

The LLM can repeatedly observe only development feedback. A program $\mathcal{Q}$ materializes $X_{\mathcal{Q}}(\mathcal{T})$, and ϕ denotes a predictor family and hyperparameters. We seek a verified pair

$$(\mathcal{Q}^\star, \phi^\star) = \arg\min_{\mathcal{Q},\phi} \; \mathcal{R}(\mathcal{Q}, \phi) + \lambda_c \mathcal{C}(\mathcal{Q}, \phi) + \lambda_\tau \log(1 + \tau(\mathcal{Q})), \tag{2}$$

where $\mathcal{R}$ is the expected test risk, $\mathcal{C}$ is a fixed description length (Equation 4), and τ is materialization time. Selection labels are used once after the candidate set is frozen; test labels are never used for search or selection.

Point-in-time correctness is defined relative to each target row rather than a global dataset date; violating it is a form of data leakage (Kaufman et al., 2012). Every event row contributing to z_i must

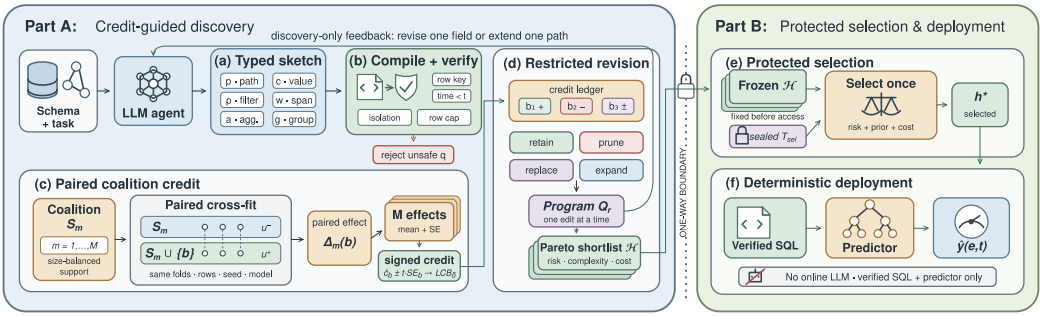

Figure 2: **RelCraft and its reliability boundary.** The LLM proposes typed sketches and predictor edits. Deterministic compilation and verification precede paired coalition evaluation. Every feature and predictor candidate is explicitly cross-fitted before it can enter the Pareto shortlist. A frozen shortlist crosses a one-way boundary for protected selection; deployment uses no LLM.

satisfy a traced source timestamp $t_x < t_i$; transformations, category vocabularies, and normalization statistics must be fitted on the training side of the current split. A valid program must also preserve exactly one feature row per target identifier. These conditions separate three objects that are often conflated: syntactic executability, temporal admissibility, and predictive utility.

4 RelCraft

Figure 2 gives an overview. The LLM proposes typed sketches and predictor edits; deterministic compilation and verification (Section 4.1) precede paired coalition credit (Section 4.2) and joint search (Section 4.3), after which a protected selector chooses the deployed pair (Section 4.4).

4.1 Typed Sketches and Verification

A feature sketch $s = (p, c, v, \rho, w, a, g)$ specifies a schema path p, cutoff column c, value v, row predicate ρ, lookback window w, aggregation a, and optional grouping g. For target row $z = (e, t)$, it denotes

$$\Phi_s(D, z) = a\big(\{v(x) : x \in J_p(D, e),\ \rho(x),\ t - w \le t_x < t\}; g\big), \tag{3}$$

where $J_p(D, e)$ is the multiset of source rows reachable from e along p and $t_x = c(x)$ is the cutoff-column timestamp of row x. The strict upper bound excludes events at or after the cutoff. Paths are bounded, acyclic sequences of declared key edges; w comes from a finite lookback set, and a is a type-compatible aggregation. The compiler follows those edges, pushes row/time filters before aggregation, groups by the evaluation `row_id`, and left-joins the result back to the target table. Empty counts are zero, other missing aggregates use a fixed token, and grouped features keep only categories learned on the training partition. Hence validation or test rows cannot change the feature schema.

Before execution, an abstract-syntax-tree verifier checks: (i) schema-resolved columns and declared keys; (ii) a strict cutoff on every event source; (iii) exactly one output row per evaluation identifier; and (iv) bounded intermediate cardinality and isolated split inputs. A zero-label dry run validates output names/types, row count, identifier order, duplicates, and finite values. The agent sees only a short code such as `MISSING_CUTOFF` or `ROW_EXPANSION`; it never sees protected rows or outcomes. Only programs satisfying every check enter model evaluation.

We charge each block a fixed structural length. For $\mathbf{s}(q) = [n_d(q), n_j(q), n_\rho(q), n_w(q), n_g(q)]^\top$,

$$\mathcal{C}(q) = 1 + \boldsymbol{\alpha}^\top \mathbf{s}(q), \qquad \boldsymbol{\alpha} = (0.25, 0.25, 0.10, 0.10, 0.25), \tag{4}$$

where n_d is path depth, n_j the number of joins, n_ρ the number of row predicates, n_w the number of non-default windows, and n_g the number of grouped outputs. Program length is additive over blocks, and predictor length adds one unit for the learner family plus one unit for each non-default discrete setting. The same definition is used by block credit, the Pareto shortlist, and the protected prior, so the search phase and final selector do not optimize different notions of complexity.

Safety scope. The verifier certifies only properties expressible in the sketch language. It covers illegal schema edges, missing or non-strict event cutoffs, row expansion, direct split or label access, and resource bounds. It does not certify that an upstream database snapshot was itself constructed without future information, that a semantically valid business field is not a target proxy, or that an external data-ingestion process preserved temporal provenance. Thus the safety claim is deliberately syntactic and operational: every generated event access must contain a strict target-relative cutoff and every accepted matrix must be row aligned, but domain-level leakage analysis remains outside the compiler. Appendix A gives the full grammar productions, verifier invariants, and threat model.

4.2 Paired Cross-Fitted Coalition Credit

Discovery uses cross-fitting (Chernozhukov et al., 2018) over fixed entity-grouped, forward-chaining folds $\mathcal{F} = \{F_k\}_{k=1}^{K_f}$. Base and augmented programs are trained on the same preceding groups and evaluated on the same held-out fold with identical preprocessing, predictor ϕ, and seed. Regression utility is $1 - \min(|y - \widehat{y}|/s_y, 1)$, with training-fold scale s_y; classification uses positive–negative concordance, whose fold average is AUROC. Both lie in $[0, 1]$.

For a program with B blocks, predecessor coalitions are drawn independently with

$$\nu_b(S) = \frac{1}{B}\binom{B-1}{|S|}^{-1}, \qquad S \subseteq \mathcal{Q} \setminus \{b\}, \tag{5}$$

which gives equal total mass to every coalition size, as in the Shapley value (Shapley, 1953). Let $\mathcal{U}_k$ be the held-out comparison units of fold k (targets for regression and positive–negative pairs for classification). If $d_i^{(k)}(b; S_m, \phi)$ is the matched utility difference on unit $i \in \mathcal{U}_k$, define

$$\begin{aligned} \Delta_m(b) &= \sum_{k=1}^{K_f} \frac{|\mathcal{U}_k|}{\sum_\ell |\mathcal{U}_\ell|} \frac{1}{|\mathcal{U}_k|} \sum_{i \in \mathcal{U}_k} d_i^{(k)}(b; S_m, \phi), \\ \widehat{c}_b &= \frac{1}{M} \sum_{m=1}^{M} \Delta_m(b) - \lambda_c \mathcal{C}(b) - \lambda_\tau \log(1 + \tau_b). \end{aligned} \tag{6}$$

Thus m indexes coalition contexts and k indexes folds; they are never interchanged. With s_Δ the standard deviation across the M contexts, search ranks blocks by $\mathrm{LCB}_\eta^{\mathrm{ctx}}(b) = \widehat{c}_b - \eta s_\Delta/\sqrt{M}$. This is deliberately a context-stability score, not a confidence interval: coalitions reuse held-out units, so s_Δ measures whether a block's sign changes across compatible subprograms. RelCraft retains a block when the lower context score is positive, prunes it when the corresponding upper score is negative, replaces unstable blocks, and expands stable path families.

Every job key hashes the snapshot, coalition, folds, preprocessing, predictor, and seed. Base and augmented programs share all of these values; a cache hit is reused only on an exact match. Static rejection creates no predictor fit. A query that passes verification but later times out receives no predictive gain and still incurs its structural and elapsed-time penalties rather than disappearing from the average. Credit is indexed by (b, ϕ) and invalidated when ϕ changes. Appendix B gives the estimator derivation and full failure/cache policy.

4.3 Joint Program–Predictor Search

The state is $h_r = (\mathcal{Q}_r, \phi_r)$. The LLM proposes typed blocks and predictor configurations from fixed menus. A feature edit is compiled and credited under the current predictor; a predictor edit is separately cross-fitted on the current program. Safe feature edits are prioritized by

$$\mathcal{A}_r(b) = \mathrm{LCB}_\eta^{\mathrm{ctx}}(b) + \kappa\sqrt{\frac{\log(1+r)}{1+n_b}} - \lambda_e \widehat{\tau}_b, \tag{7}$$

where n_b counts evaluations of the block's path family. Only measured feature/program pairs and measured model/program pairs enter the visited set. Algorithm 1 summarizes the procedure; Appendix B gives complete pseudocode and accounting.

Algorithm 1 RelCraft search and selection.

Require: $D, G_D, \mathcal{T}_{\mathrm{tr/dev/sel}}$, rounds R, coalitions M

$h_0 = (\mathcal{Q}_0, \phi_0)$; $\mathcal{V} \leftarrow \{h_0\}$; ledger $\mathcal{L} \leftarrow \varnothing$
for $r = 1, \ldots, R$ **do**
  $(\mathcal{B}_r, \mathcal{M}_r) \leftarrow$ AGENTPROPOSE$(h_{r-1}, \mathcal{L})$
  for $b \in \mathcal{B}_r$ **do**
    compile/verify b; if safe, credit and cross-fit $(\mathcal{Q}_{r-1} \cup \{b\}, \phi_{r-1})$
  end for
  for $\phi \in \mathcal{M}_r$ **do**
    cross-fit $(\mathcal{Q}_{r-1}, \phi)$; add its measured risk to $\mathcal{V}$
  end for
  $h_r \leftarrow$ AGENTEDIT using only evaluated edits; verify and cross-fit h_r
  add h_r to $\mathcal{V}$; invalidate credits indexed by $\phi \neq \phi_r$
end for
freeze $\mathcal{H} \leftarrow \mathrm{Pareto}(\mathcal{V}; \widehat{\mathcal{R}}_{\mathrm{dev}}, \mathcal{C}, \tau)$
return protected choice $h^\star$ from Equation 8

Search stops at 30 agent-visible validations, 240 unique cross-fit jobs, 60 minutes, 60 turns, or two rounds without a positive context score. Every model-only cross-fit and the final assembled-program evaluation count toward the same fit budget.

The frozen shortlist is the non-dominated subset of visited pairs under development risk, description length, and runtime:

$$\mathcal{H} = \mathrm{Pareto}\{(\widehat{\mathcal{R}}_{\mathrm{dev}}(h), \mathcal{C}(h), \tau(h)) : h \in \mathcal{V}\}.$$

This removes candidates that are no better on any selection-relevant axis and prevents repeated protected evaluation of near-duplicate programs. Crucially, an assembled feature–predictor combination is cross-fitted as that exact pair before it can enter $\mathcal{V}$ or $\mathcal{H}$.

4.4 PROTECTED COMPLEXITY-CORRECTED SELECTION

Before search, the bounded grammar and predictor menu define a fixed universe $\mathcal{H}_0$. We assign the fixed prior $\pi_0(h) = \exp[-\gamma \mathcal{C}(h)] / \sum_{g \in \mathcal{H}_0} \exp[-\gamma \mathcal{C}(g)]$. Protected targets are grouped by entity, and selection uses a loss in $[0, 1]$: group-mean Brier loss for classification and clipped, training-scale-normalized absolute error for regression. With n_{sel} independent groups and a predeclared confidence level $\delta \in (0, 1)$,

$$\rho_\delta(h) = \sqrt{\frac{\log \pi_0(h)^{-1} + \log \delta^{-1}}{2 n_{\mathrm{sel}}}}, \qquad h^\star = \arg\min_{h \in \mathcal{H}} \{\widehat{\mathcal{R}}_{\mathrm{sel}}(h) + \rho_\delta(h) + \lambda_\tau \log(1 + \tau(h))\}. \tag{8}$$

The harness evaluates each frozen candidate once and returns no protected outcome to the agent.

Theorem 1 (Protected selection)**.** *If protected entity-group losses lie in* $[0, 1]$*, are mutually independent, and are independent of the adaptive construction of* $\mathcal{H}$*, then with probability at least* $1 - \delta$*, simultaneously for all* $h \in \mathcal{H}$*,*

$$\mathcal{R}_{\mathrm{bnd}}(h) \leq \widehat{\mathcal{R}}_{\mathrm{sel}}(h) + \rho_\delta(h), \tag{9}$$

where $\mathcal{R}_{\mathrm{bnd}}(h)$ *is the expected bounded selection loss of* h*.*

This is a guarantee for the bounded selector loss, not for AUROC or raw MAE; those remain test metrics. The proof, assumptions, protected-size calculation, and refit boundary are in Appendix C.

Proof sketch. Condition on the discovery transcript, which fixes $\mathcal{H}$ independently of protected outcomes. Hoeffding's inequality (Hoeffding, 1963) for a fixed h with failure probability $\delta \pi_0(h)$ gives Equation 9; summing over $\mathcal{H} \subseteq \mathcal{H}_0$ costs at most $\delta \sum_h \pi_0(h) \leq \delta$. Consequently the result holds for the protected choice even though that choice depends on protected losses. A desired slack ϵ requires $n_{\mathrm{sel}} \geq [\log \pi_0(h)^{-1} + \log \delta^{-1}]/(2\epsilon^2)$ independent entity groups. For reported test evaluation, each candidate is a deterministic pipeline fitted on $\mathcal{T}_{\mathrm{tr}} \cup \mathcal{T}_{\mathrm{dev}}$, and the selected fitted artifact is carried unchanged to test; a refit with protected labels would be a new candidate outside the guarantee. □

4.5 Inference Boundary

After selection, the agent, credit ledger, development diagnostics, and rejected candidates are discarded. For target (e, t), deployment computes

$$\widehat{y}(e,t) = f_{\theta^\star}(\Phi_{\mathcal{Q}^\star}(D, e, t)), \qquad \mathsf{Safe}(q) = 1 \ \ \forall q \in \mathcal{Q}^\star, \tag{10}$$

where $f_{\theta^\star}$ is the selected predictor $\phi^\star$ fitted on $\mathcal{T}_{\mathrm{tr}} \cup \mathcal{T}_{\mathrm{dev}}$ and Safe is the verifier predicate of Appendix A. The stored artifact contains the verified SQL hashes, preprocessing state, fitted predictor, and cutoff certificate. It does not contain an LLM call or a search-time evaluator. This boundary is operational as well as conceptual: online latency in Table 3 includes feature retrieval, selected SQL, preprocessing, and prediction, but excludes all discovery and coalition work.

5 Experiments

5.1 Protocol

We evaluate all license-compatible entity-prediction tasks from RelBenchV1 (RB-V1) (Robinson et al., 2024), RelBenchV2 (RB-V2) (Gu et al., 2026), and 4DBInfer (Wang et al., 2024): 21 classification and 11 regression tasks. Classification uses test AUROC (%). Regression uses normalized MAE (NMAE): each task's MAE is divided by the corresponding raw-table LightGBM MAE, so 1.0 represents that task-specific reference rather than a shared error scale. Suite means weight tasks equally, and classification and regression are never averaged into one headline number.

Appendix Table 8 lists which published baselines cover each suite. The Coverage column of Table 2 reports the number of tasks contributing to each published aggregate, while all controlled system metrics use the same complete 32-task set. Latency is measured end to end per 10K target rows after a warm-up execution; timeouts remain failures rather than censored fast runs.

Controlled agentic runs use GPT-5.2 at temperature 1, at most 8,000 output tokens per turn, five rollout seeds, five forward-chaining folds, and $M = 8$ coalitions. All methods share snapshots, split rows, predictor menu, seeds, DuckDB (Raasveldt & Mühleisen, 2019), a 900-second query timeout, and the stopping limits above. The protected split is the final 25% of the original validation period. A "Fit" is one unique cross-fit job for a fixed program–predictor–seed key and internally trains five fold models. We fix $(\lambda_c, \lambda_\tau, \lambda_e, \eta, \kappa, \gamma) = (10^{-3}, 10^{-3}, 10^{-3}, 1, 10^{-2}, 1)$ in normalized utility/cost units before benchmark evaluation. Appendix D gives the predictor grid, normalization, aggregation, coverage, confidence-interval calculation, and complete robustness protocol.

Table 1 is the complete family-level grid over LightGBM (Ke et al., 2017), XGBoost (Chen & Guestrin, 2016), random forests (Breiman, 2001), and CatBoost (Prokhorenkova et al., 2018); Appendix D specifies leaves, subsampling, minimum leaf size, preprocessing, and fold-local early stopping. Model proposals select from this finite grid rather than inventing learner code. All category vocabularies, imputation values, scales, and target transforms are re-estimated on the training side of each fold.

Table 1: Fixed predictor menu. Regression additionally chooses squared, absolute, or Huber objective and identity/`log1p` target transform when valid.

Family	Variant/depth	LR	Trees/iters.
LightGBM	GBDT/DART/GOSS; $-1/8$	.03/.10	300/600
XGBoost	tree/DART; $4/8$	.03/.10	300/600
Random forest	depth $\infty/12$	–	300/600
CatBoost	depth $6/8$	.03/.10	300/600

We keep two evidence tracks separate. Table 2 uses complete published aggregates, including DFS (Kanter & Veeramachaneni, 2015) and GraphSAGE (Hamilton et al., 2017), for benchmark positioning; it is not a paired compute comparison. Table 3 uses new reruns in a common evaluation harness of RelCraft, RelAgent, RelAgent+guard, and ReFuGe under the matched resources above. Incomplete published methods are retained only where their suite coverage is explicit and are excluded from full-coverage paired claims. This prevents a missing task or a stronger external predictor budget from being interpreted as evidence about RelCraft's search mechanism.

5.2 Predictive Quality and Statistical Scope

Table 2 reports benchmark positioning; controlled mechanism tests follow in Sections 5.3 and 5.4. The published RelAgent row is not a compute-matched rerun, so we treat the differences as descrip-

Table 2: **Predictive performance.** Classification reports AUROC (%); regression reports per-task MAE normalized by raw-table LightGBM. Non-RelCraft rows are complete published aggregates; "–" means no complete suite result. RelCraft reports mean±standard deviation over five rollout seeds. Bold: best; underline: second best.

	Classification AUROC ↑			Regression normalized MAE ↓		Coverage
Method	RB-V1	RB-V2	4DBInfer	RB-V1	RB-V2	Tasks
LightGBM / XGBoost	63.66	76.78	50.10	1.000	1.000	32/32
DFS+GBDT	75.62	–	74.35	0.879	–	26/32
GraphSAGE	75.83	83.99	79.40	0.918	0.626	32/32
RelGNN	78.06	–	–	0.861	–	21/32
RDBLearn	75.55	–	78.60	0.906	–	26/32
KumoRFM-v2	**79.60**	85.91	79.96	0.822	0.601	32/32
RelAgent	78.38	<u>87.32</u>	<u>81.38</u>	<u>0.817</u>	<u>0.548</u>	32/32
RelCraft	<u>79.34</u>±0.22	**88.14**±0.19	**82.26**±0.25	**0.793**±0.006	**0.528**±0.004	32/32

Table 3: **Controlled diagnostics over 32 tasks.** AUTC is absolute area under the best development-utility–versus–fit curve; Val. is the agent-visible validations needed to recover 95% of the final gain; Fits counts unique five-fold cross-fit jobs. Dev/Test and Sel/Test are absolute metric gaps in points. Baseline Sel/Test is a post-hoc frozen-candidate diagnostic, not the RelCraft selector. Latency is seconds per 10K rows.

	Search			Reliability			Deployment		
Method	AUTC↑	Val.↓	Fits↓	SQL fail (%)↓	Dev/Test↓	Sel/Test↓	σ_{roll} ↓	#Blocks↓	Latency↓
ReFuGe	0.742	30.0	150.0	9.7	1.67	1.49	1.18	54.3	9.1
RelAgent	0.781	27.8	**139.0**	6.8	1.54	1.31	0.91	46.8	7.4
RelAgent+guard	0.786	26.9	145.0	4.9	1.42	1.18	0.84	44.5	7.1
RelCraft	**0.814**	**21.6**	211.4	**1.9**	**1.02**	**0.56**	**0.51**	**31.6**	**5.8**

tive: relative to it, RelCraft is higher in classification AUROC by 0.96, 0.82, and 0.88 points across the three suites and lowers NMAE from 0.817 to 0.793 and from 0.548 to 0.528. It remains 0.26 points below KumoRFM-v2 on RB-V1.

From the reported five-seed standard deviations, two-sided 95% t_4 intervals for RelCraft are $[79.07, 79.61]$, $[87.90, 88.38]$, $[81.95, 82.57]$, $[0.786, 0.800]$, and $[0.523, 0.533]$ in table order (Table 10). They quantify rollout stability, not significance against unpaired published aggregates. Controlled comparisons use the task as the inferential unit; Appendix D specifies the paired task bootstrap and randomization protocol and avoids row-level pseudo-replication.

5.3 Search, Reliability, and Cost

For controlled diagnostics, V is AUROC or $1 - \text{NMAE}$ on the 0–1 scale. With h_{dev} the candidate a method selects from development feedback and $h^\star$ the protected choice, Dev/Test and Sel/Test are $100|V_{\text{dev}}(h_{\text{dev}}) - V_{\text{te}}(h_{\text{dev}})|$ and $100|V_{\text{sel}}(h^\star) - V_{\text{te}}(h^\star)|$ in metric points. For a common budget of $N = 240$ unique cross-fit jobs, we define

$$\text{AUTC}_N = \frac{1}{N} \sum_{j=1}^{N} \max_{k \le j} V_{\text{dev}}(h_k), \tag{11}$$

where h_k is the best valid state available after job k. Val. is the first agent-visible validation that recovers 95% of the final *gain over the initial program*. AUTC, validations, logical cross-fit jobs, and five-fold atomic model fits thus measure different aspects of cost.

The running maximum in AUTC ensures that a later failed proposal cannot erase an earlier valid program. If a method terminates before the common budget, its best valid value is carried forward, so early termination is neither dropped nor rewarded with a shorter averaging interval. Cross-rollout deviation is computed from each seed's final selected utility after task normalization. The gap met-

rics compare the same frozen candidate on two chronological slices and therefore measure estimate calibration, not regret to an unavailable test oracle.

Table 3 shows the intended trade-off. RelCraft uses 211.4 logical cross-fit jobs versus 139.0 for RelAgent—equivalently, 1,057 versus 695 atomic fold-model trainings on average—but needs fewer agent-visible validations to recover 95% of its final gain (21.6 versus 27.8) and raises AUTC from 0.781 to 0.814. We therefore claim improved *decision and fit-trajectory quality*, not lower total training compute; all methods remain under the matched 60-minute/240-job caps.

Typed compilation reduces SQL failure from 6.8% for RelAgent and 4.9% for its prompt-guard variant to 1.9%. RelCraft also reduces the descriptive protected–test gap from 1.31 to 0.56, rollout deviation from 0.91 to 0.51, selected blocks from 46.8 to 31.6, and latency from 7.4 to 5.8. Appendix B bounds the number of requested discovery jobs, Appendix D gives exact metric formulas, and Table 11 lists all relative changes.

5.4 Ablations and Controlled Probes

Table 4 assigns a separate observable to each mechanism. Removing the grammar raises SQL failures to 8.7%; removing coalition credit saves internal jobs but loses 0.65 AUROC points and retains 10.5 more blocks; leave-one-out credit only partly recovers this loss. Removing protected selection yields the largest gap (1.41 versus 0.56) while changing AUROC by only -0.29 points, indicating that the protected boundary mainly improves selection calibration rather than mean accuracy. Removing the complexity or execution term expands the deployed program by 57.6% or 49.4%. The execution-cost ablation has slightly higher raw AUROC (79.36 versus 79.34) but 47.2 rather than 31.6 blocks, confirming an accuracy–deployability trade-off rather than uniform metric dominance.

Table 4: **Component ablation.** AUROC is RB-V1 classification (%); other diagnostics use the full 32-task controlled protocol. Gap is the protected–test absolute gap in points; SQL is the SQL failure rate.

Variant	AUROC↑	Gap↓	SQL (%)↓	Fits↓	#Blocks↓
RelCraft	79.34	**0.56**	**1.9**	211.4	**31.6**
w/o typed grammar	78.72	0.64	8.7	224.8	36.9
w/o static verifier	79.18	0.67	2.4	218.7	33.1
w/o coalition credit	78.69	0.91	5.9	**139.0**	42.1
leave-one-out credit	78.91	0.83	3.1	167.6	38.7
w/o protected split	79.05	1.41	2.0	207.9	32.4
w/o complexity prior	79.31	0.65	2.1	214.6	49.8
w/o execution cost	**79.36**	0.63	2.0	216.1	47.2

The two safety ablations separate syntax control from temporal reasoning. Prompts without the grammar can propose a broader invalid surface, whereas removing only the static verifier leaves typed generation intact but allows more unsafe assemblies to reach execution. Similarly, scalar feedback (w/o coalition credit) and leave-one-out feedback have different budgets: the former performs the fewest internal jobs, while the latter adds paired evaluation in only the current full-program context.

Table 5 isolates expressivity, interaction credit, and the declared leakage threat model. Triangle requires a typed two-hop motif; XOR contains two individually uninformative aggregates whose joint parity is predictive; Leak/2K is a fixed bank of 2,000 sketches with explicit post-cutoff predicates. Typed compilation admits zero such future-edge sketches, and coalition credit reaches 1.000 XOR AUROC and 0.914 Sign-F1. These are stress tests, not a universal claim against undeclared snapshot

Table 5: **Controlled probes.** Triangle/XOR report AUROC (0–1 scale); Leak/2K counts admitted future-edge sketches; Sign-F1 compares credit signs with exhaustive coalition effects.

Variant	Triangle↑	XOR↑	Leak/2K↓	Sign-F1↑
Free SQL + scalar	0.842	0.500	187	0.508
Typed + scalar	0.996	0.500	0	0.544
Typed + LOO	0.998	0.500	0	0.731
RelCraft	**1.000**	**1.000**	**0**	**0.914**

Table 6: **Backbone robustness.** AUROC (%) averages 21 classification tasks; Rec@10 is final gain over the initial program recovered after ten turns.

	Average AUROC		Rec@10 (%)	
Backbone	RelAgent	RelCraft	RelAgent	RelCraft
GPT-5.2	80.79	**81.71**	84.8	**92.6**
DeepSeek-V4	80.22	**81.24**	82.6	**91.2**
Qwen3-32B	79.33	**81.00**	80.4	**89.8**
Mean	80.11	**81.32**	82.6	**91.2**

or target leakage. The ordering of the ablation AUROC and probe Sign-F1 values is consistent with, but does not by itself prove, the value of multi-context interaction credit.

Across GPT-5.2, DeepSeek-V4 (DeepSeek-AI, 2026), and Qwen3-32B (Yang et al., 2025), RelCraft improves every backbone, averages 81.32 versus 80.11 AUROC, and recovers 91.2% versus 82.6% of final gain by turn ten (Table 6). Its cross-backbone AUROC range is 0.71 points, compared with 1.46 for RelAgent. Appendix D fixes the full predictor, split, and early-budget protocol used for this comparison.

These experiments answer different questions, and each claim is tied to one designated evidence track (Table 12). Published aggregates support only predictive positioning; common-harness reruns support search, reliability, and deployment comparisons; within-RelCraft ablations test individual mechanisms; and the synthetic probes test declared failure modes. In particular, the smaller protected–test gap is a calibration diagnostic, whereas evidence specific to the selector comes from removing the protected split in Table 4. We do not treat agreement across these views as additional independent samples.

6 Scope and Limitations

RelCraft addresses reliability within a declared program language, not arbitrary database leakage. The safety scope in Section 4.1 applies: the verifier cannot recover provenance omitted from an upstream snapshot or determine that an allowed business field is a semantic target proxy, so the zero-violation probe should be read within its explicit-future-edge generator.

Coalition feedback is also a search heuristic rather than a causal estimand. Pairing removes fold, example, preprocessing, and seed differences between the base and augmented program, while multiple contexts expose feature interactions. Nevertheless, $M = 8$ contexts leave Monte Carlo variation, and $\mathrm{LCB}^{\mathrm{ctx}}$ is intentionally not reported as a confidence interval. The method pays for this information: Table 3 shows 52% more logical cross-fit jobs than RelAgent, even though it reduces agent-visible decisions and obtains a better fit trajectory.

Finally, Theorem 1 controls only the bounded group loss used behind the protected boundary and assumes independent entity groups. It does not turn the observed AUROC/MAE differences into theoretical guarantees. The main predictive comparison also mixes new RelCraft runs with published complete-suite aggregates, so its sub-point gains are descriptive; mechanism claims instead rely on common-harness diagnostics and within-system ablations. These boundaries narrow the claim but make each reported conclusion traceable to a corresponding experiment.

7 Conclusion and Outlook

RelCraft treats relational feature engineering as a reliability problem by separating LLM proposal, deterministic compilation, block-level credit, and protected selection. Typed sketches enforce declared schema paths and target-relative cutoffs, coalition evaluation provides context-aware block feedback, and the protected selector chooses among frozen SQL–predictor pairs without returning outcomes to the agent. Across 32 tasks, RelCraft remains competitive with the strongest published baselines and, in matched reruns, improves search efficiency (AUTC 0.781 to 0.814), explicit future-edge rejection, stability, compactness (46.8 to 31.6 blocks), and deployment latency, at the cost of 52% more offline cross-fit jobs. At inference time, the system reduces to verified SQL, fixed preprocessing, and a conventional predictor, with no LLM in the serving path.

Looking forward, RelCraft could be extended with richer temporal operators, lineage certificates, and domain-specific proxy checks to narrow the gap between syntactic point-in-time safety and full data-provenance assurance. Coalition credit could be made cheaper through adaptive context allocation, early stopping, or surrogates, while the protected boundary could be adapted to streaming or periodically refreshed databases. More broadly, RelCraft suggests that reliable LLM-based data agents should combine constrained program languages, auditable execution, and statistically disciplined selection.

AI USE STATEMENT

[TODO] In this work, we used generative AI tools for [tasks with required disclosure]. We have not used generative AI tools for [other tasks with required disclosure], and [the rest of the required disclosure tasks] are not applicable to this work. We have reviewed all AI-assisted work and take responsibility for the final content of this work.

ETHICS STATEMENT

All experiments use public relational benchmark databases. Reliable point-in-time feature construction can reduce silent temporal leakage and make relational prediction pipelines easier to inspect. However, automated feature search may also amplify biases, privacy risks, or proxy discrimination already present in a relational database, so deployment still requires data-governance review, access control, provenance auditing, and task-specific fairness evaluation (Appendix F).

REPRODUCIBILITY STATEMENT

The typed language, verifier checks, and compilation steps are specified in Appendix A; the coalition estimator, complete search procedure, and fit accounting in Appendix B; and the full proof of Theorem 1 in Appendix C. Appendix D fixes the benchmark splits, predictor grid, coefficients, metric definitions, uncertainty calculation, and task-level resampling protocol. The reporting unit is a task–seed run: seeds are averaged within each task before tasks receive equal suite weight, and missing or timed-out runs retain the predeclared raw-table baseline value. Each run manifest stores snapshot and split hashes, LLM settings, sketch and SQL hashes, verifier certificates, coalitions, predictor configurations and seeds, preprocessing hashes, out-of-fold predictions, cache status, logical and atomic fit counts, timing, and errors. All reported metrics are regenerated from stored out-of-fold predictions, and protected selection can be replayed without an LLM.

REFERENCES

Nikhil Abhyankar, Parshin Shojaee, and Chandan K. Reddy. LLM-FE: Automated feature engineering for tabular data with LLMs as evolutionary optimizers. *Transactions on Machine Learning Research*, 2026. URL https://openreview.net/forum?id=qvI35hkpOO.

B. Aditya, Gaurav Bhalotia, Soumen Chakrabarti, Arvind Hulgeri, Charuta Nakhe, Parag, and S. Sudarshan. BANKS: Browsing and keyword searching in relational databases. In Philip A. Bernstein, Yannis E. Ioannidis, Raghu Ramakrishnan, and Dimitris Papadias (eds.), *VLDB '02: Proceedings of the 28th International Conference on Very Large Databases*, pp. 1083–1086. Morgan Kaufmann, San Francisco, 2002. ISBN 978-1-55860-869-6. doi: https://doi.org/10.1016/B978-155860869-6/50114-1. URL https://www.sciencedirect.com/science/article/pii/B9781558608696501141.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Tianlang Chen, Charilaos Kanatsoulis, and Jure Leskovec. RelGNN: Composite message passing for relational deep learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 8296–8312. PMLR, 2025. URL https://proceedings.mlr.press/v267/chen25ad.html.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016.

Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal*, 21(1):C1–C68, 2018.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence, 2026.

Vijay Prakash Dwivedi, Sri Jaladi, Yangyi Shen, Federico López, Charilaos I. Kanatsoulis, Rishi Puri, Matthias Fey, and Jure Leskovec. Relational graph transformer. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=2d3j6bt21A.

Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. The reusable holdout: Preserving validity in adaptive data analysis. *Science*, 349(6248):636–638, 2015.

Matthias Fey, Weihua Hu, Kexin Huang, Jan Eric Lenssen, Rishabh Ranjan, Joshua Robinson, Rex Ying, Jiaxuan You, and Jure Leskovec. Position: Relational deep learning – graph representation learning on relational databases. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 13592–13607. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/fey24a.html.

Justin Gu, Rishabh Ranjan, Charilaos Kanatsoulis, Haiming Tang, Martin Jurkovic, Valter Hudovernik, Mark Znidar, Pranshu Chaturvedi, Parth Shroff, Fengyu Li, et al. RelBench v2: A large-scale benchmark and repository for relational data. *arXiv preprint arXiv:2602.12606*, 2026.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 1025–1035, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 978-1-5108-6096-4. doi: 10.5555/3294771.3294869.

Sungwon Han, Jinsung Yoon, Sercan O. Arik, and Tomas Pfister. Large language models can automatically engineer features for few-shot tabular learning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 17454–17479. PMLR, 2024. URL https://proceedings.mlr.press/v235/han24f.html.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

Noah Hollmann, Samuel Müller, and Frank Hutter. Large language models for automated data science: Introducing CAAFE for context-aware automated feature engineering. In *Advances in Neural Information Processing Systems*, volume 36, pp. 44753–44775. Curran Associates, Inc., 2023. doi: 10.52202/075280-1938. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/8c2df4c35cdbee764ebb9e9d0acd5197-Abstract-Conference.html.

Xingyue Huang, Louis Tichelman, Jinwoo Kim, Krzysztof Olejniczak, and İsmail İlkan Ceylan. RelAgent: LLM agents as data scientists for relational learning. *arXiv preprint arXiv:2605.07840*, 2026. URL https://arxiv.org/abs/2605.07840.

Valter Hudovernik, Federico López, Vid Kocijan, Akihiro Nitta, Jan Eric Lenssen, Jure Leskovec, and Matthias Fey. KumoRFM-2: Scaling foundation models for relational learning. *arXiv preprint arXiv:2604.12596*, 2026. doi: 10.48550/arXiv.2604.12596. URL https://arxiv.org/abs/2604.12596.

James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 1–10, 2015.

Shachar Kaufman, Saharon Rosset, Claudia Perlich, and Ori Stitelman. Leakage in data mining: Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data*, 6(4):15:1–15:21, 2012.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Chohee Kim, Mihaela Van Der Schaar, and Changhee Lee. Discovering features with synergistic interactions in multiple views. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 24562–24583. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/kim24ag.html.

Heegyu Kim, Jeon Taeyang, SeungHwan Choi, Seungtaek Choi, and Hyunsouk Cho. FLEX: Expert-level false-less EXecution metric for text-to-SQL benchmark. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4448–4475, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.228. URL https://aclanthology.org/2025.naacl-long.228/.

Kyungho Kim, Geon Lee, Juyeon Kim, Dongwon Choi, Shinhwan Kang, and Kijung Shin. ReFuGe: Feature generation for prediction tasks on relational databases with LLM agents. In *Proceedings of the ACM Web Conference 2026*, pp. 8501–8504. Association for Computing Machinery, 2026. doi: 10.1145/3774904.3792895. URL https://doi.org/10.1145/3774904.3792895.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Jaehyun Nam, Kyuyoung Kim, Seunghyuk Oh, Jihoon Tack, Jaehyung Kim, and Jinwoo Shin. Optimized feature generation for tabular data via LLMs with decision tree reasoning. In *Advances in Neural Information Processing Systems*, volume 37, pp. 92352–92380. Curran Associates, Inc., 2024. doi: 10.52202/079017-2932. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/a7ebe2e8d8cfd2fcec6cd77f9e6fd34d-Abstract-Conference.html.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. CatBoost: Unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Mark Raasveldt and Hannes Mühleisen. DuckDB: An embeddable analytical database. In *Proceedings of the 2019 International Conference on Management of Data (SIGMOD)*, pp. 1981–1984, 2019.

Rishabh Ranjan, Valter Hudovernik, Mark Znidar, Charilaos I. Kanatsoulis, Roshan Reddy Upendra, Mahmoud Mohammadi, Joe Meyer, Tom Palczewski, Carlos Guestrin, and Jure Leskovec. Relational transformer: Toward zero-shot foundation models for relational data. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=rpPtgMC5s9.

Joshua Robinson, Rishabh Ranjan, Weihua Hu, Kexin Huang, Jiaqi Han, Alejandro Dobles, Matthias Fey, Jan E. Lenssen, Yiwen Yuan, Zecheng Zhang, Xinwei He, and Jure Leskovec. RelBench: A benchmark for deep learning on relational databases. In *Advances in Neural Information Processing Systems*, volume 37, pp. 21330–21341. Curran Associates, Inc., 2024. doi: 10.52202/079017-0672. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/25cd345233c65fac1fec0ce61d0f7836-Abstract-Datasets_and_Benchmarks_Track.html.

Lloyd S. Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker (eds.), *Contributions to the Theory of Games II*, volume 28 of *Annals of Mathematics Studies*, pp. 307–317. Princeton University Press, 1953.

Minjie Wang, Quan Gan, David Wipf, Zhenkun Cai, Ning Li, Jianheng Tang, Yanlin Zhang, Zizhao Zhang, Zunyao Mao, Yakun Song, et al. 4DBInfer: A 4D benchmarking toolbox for graph-centric predictive modeling on RDBs. *Proceedings of the VLDB Endowment. ISSN*, 2150:8097, 2024.

Yanbo Wang, Xiyuan Wang, Quan Gan, Minjie Wang, Qibin Yang, David Wipf, and Muhan Zhang. Griffin: Towards a graph-centric relational database foundation model. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 64604–64627. PMLR, 2025. URL https://proceedings.mlr.press/v267/wang25da.html.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. arXiv preprint arXiv:2505.09388, May 2025.

Yanlin Zhang, Linjie Xu, Quan Gan, David Wipf, and Minjie Wang. RDBLearn: Simple in-context prediction over relational databases. *arXiv preprint arXiv:2602.18495*, 2026. doi: 10.48550/arXiv.2602.18495. URL https://arxiv.org/abs/2602.18495.

Ruiqi Zhong, Tao Yu, and Dan Klein. Semantic evaluation for text-to-SQL with distilled test suites. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 396–411, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.29. URL https://aclanthology.org/2020.emnlp-main.29/.

Roadmap. Appendix A makes the typed language, compiler invariants, structural code, and safety scope explicit. Appendix B gives the paired estimator, the closed feature–predictor search, and cost accounting. Appendix C states and proves the protected-selection guarantee with an operational finite prior. Appendix D fixes the experimental protocol, metric denominators, uncertainty calculation, robustness tests, failure audit, and a claim–evidence map; Appendices E and F discuss limitations and broader impacts. All additional numerical comparisons below are algebraic summaries of the tables in the main paper; no unreported task-level outcomes are introduced.

A Typed Program Language and Verifier

A.1 Relational Sketch Grammar

RelCraft never asks the LLM to emit executable SQL. It emits a typed sketch

$$s = (p, c, v, \rho, w, a, g),$$

whose fields have the domains in Table 7.

Table 7: Sketch fields. Schema fields and predicate constants are drawn from the finite, training-only task dictionary frozen before discovery. Together with the depth and program caps, these domains induce the fixed candidate language used by protected selection.

Field	Type	Allowed values
p	path	declared acyclic PK–FK path, depth ≤ 4
c	time	event timestamp
v	value	numeric, Boolean, category, or indicator
ρ	predicate	bounded conjunction of typed source comparisons
w	window	$\{1, 7, 30, 90, 180, 365, \infty\}$ days
a	aggregation	count, sum, mean, min, max, std, last
g	grouping	none or one bounded category

Paths are enumerated from the target entity through declared key edges. A path cannot revisit a table alias, traverse an undeclared equality, or exceed the depth-four cap. Before any rollout, every predicate-eligible column receives a finite dictionary $\mathcal{D}_j$ computed from training-side metadata only; the dictionary values and hash are serialized in the task manifest. Predicates can compare a source column only with a type-compatible member of $\mathcal{D}_j$ and cannot introduce a free-form literal. The number of conjuncts and grouped outputs is bounded by the declared structural cap. Arbitrary subqueries, user-defined functions, free-form joins, and references to workspace tables are outside the language. Consequently the block dictionary is finite before development or protected outcomes are observed.

For a target $z = (e, t)$, let $J_p(D, e)$ be the multiset of source rows reachable from e along p. The denotation is

$$\Phi_s(D, z) = a\left(\left\{ v(x) : \begin{array}{l} x \in J_p(D, e),\ \rho(x), \\ t - w \leq t_x < t \end{array} \right\}; g\right). \tag{12}$$

Here $v(x)$ is the aggregated value and c is only the cutoff-column field. The strict inequality is part of the grammar, not a prompt convention, and it is required for every event-bearing table on the path rather than only the terminal table. Type checking permits count for any source, sum/mean/std only for numeric values, last only with an event time, and min/max only for ordered types. For a static dimension table with no event time, the data snapshot identifier must certify that the row existed before the target period. Grouped features retain the top-k categories estimated on the corresponding training partition and map all others to `OTHER`. Empty counts are zero; other empty aggregates use one fixed missing token. Dictionaries, top-k vocabularies, missing-value state, and all other preprocessing choices are re-fitted independently inside each discovery fold.

A.2 Compilation

Compilation is deterministic:

1. Anchor the query on the split-specific evaluation table and preserve its `row_id`.
2. Materialize every path edge as a declared key join with unique aliases.
3. Push row predicates and the strict cutoff into the earliest legal common-table expression.
4. Aggregate by `row_id`; for grouped features, pivot only the training-derived vocabulary.
5. Left join the aggregate back to the evaluation table so every target appears once and in the original order.

The compiler records an abstract syntax tree, resolved schema identifiers, a normalized SQL string, and a content hash. A change to any sketch field creates a new hash and therefore a new cache key.

A.3 Static and Dynamic Verification

For compiled query q, the verifier evaluates

$$\mathsf{Safe}(q) = \mathbf{1}[A(q) \wedge R(q) \wedge P(q) \wedge I(q) \wedge B(q)], \tag{13}$$

with five independently logged checks.

- A (*anchoring*) requires the evaluation table and `row_id` to be the query root.
- R (*row alignment*) requires all non-key output expressions to be grouped or aggregated and forbids many-to-many expansion.
- P (*point in time*) traces every event-bearing source to a strict `source_time < cutoff` predicate and checks the declared lookback lower bound.
- I (*isolation*) rejects labels, protected/test materializations, workspace tables, split-dependent vocabularies, and preprocessing statistics not created from the training side of the current fold.
- B (*boundedness*) uses schema cardinalities and database statistics to reject an estimated intermediate row count above the task cap.

An accepted query then receives a zero-label dry run. It must return the expected schema, the same number and order of `row_id`s as the evaluation table, no duplicate identifiers, finite numeric values after declared missing-value handling, and no undeclared output columns. The verifier returns only an error code to the LLM: `UNKNOWN_COLUMN`, `ILLEGAL_EDGE`, `MISSING_CUTOFF`, `NONSTRICT_CUTOFF`, `ROW_EXPANSION`, `SPLIT_ACCESS`, or `RESOURCE_CAP`.

A.4 Structural Code and Threat Model

For the descriptor used in the main paper,

$$\mathbf{s}(q) = [n_d(q), n_j(q), n_\rho(q), n_w(q), n_g(q)]^\top,$$

the fixed block length is

$$\mathcal{C}(q) = 1 + \boldsymbol{\alpha}^\top \mathbf{s}(q), \quad \boldsymbol{\alpha} = (0.25, 0.25, 0.10, 0.10, 0.25), \tag{14}$$

and $\mathcal{C}(\mathcal{Q}) = \sum_{q \in \mathcal{Q}} \mathcal{C}(q)$. Here n_d is path depth; n_j, n_ρ, n_w, n_g count joins, row predicates, non-default windows, and grouped outputs. Predictor code length adds one unit for the learner family and one unit for each non-default discrete setting. The same code is used in block credit, Pareto shortlisting, and the protected prior; complexity is therefore not redefined after search.

The declared safety claim covers illegal schema edges, missing or non-strict event cutoffs, row expansion, direct split/label access, and unbounded intermediates expressible in the sketch language. Static dimension tables without timestamps are allowed only when the benchmark snapshot certifies their pre-target availability; otherwise a traced timestamp and strict cutoff are mandatory. Fold-local categorical vocabularies prevent validation or test rows from changing the feature schema.

The verifier does *not* prove that an upstream snapshot was constructed without future information, that a semantically permissible source field is not a target proxy, or that external ingestion preserved temporal provenance. The Leak/2K probe therefore tests explicit post-cutoff predicates within the declared threat model, not every possible form of database leakage.

B Coalition Credit and Joint Search

B.1 Forward-Chaining Cross-Fitting

Discovery targets are partitioned by entity into $K_f = 5$ held-out folds $\mathcal{F} = \{F_k\}_{k=1}^{K_f}$. Repeated targets for one entity remain in one group. For held-out fold F_k, the auxiliary training set $\mathcal{A}_k \subset \mathcal{T}_{\text{dev}}$ contains only groups whose latest cutoff precedes the earliest cutoff in F_k. Both sides of a comparison train on $\mathcal{T}_{\text{tr}} \cup \mathcal{A}_k$ with identical preprocessing, learner, hyperparameters, and random seed.

For regression, the unit utility is

$$u_i(\widehat{y}_i) = 1 - \min\{|y_i - \widehat{y}_i|/s_{y,k}, 1\},$$

where $s_{y,k}$ is the mean absolute deviation of the training labels from their training-fold median. For binary classification, a conceptual positive–negative pair (i, j) has

$$u_{ij} = \mathbf{1}[\widehat{y}_i > \widehat{y}_j] + \tfrac{1}{2}\mathbf{1}[\widehat{y}_i = \widehat{y}_j].$$

Its average over all cross-class pairs equals AUROC. The implementation computes the same quantity from ranks rather than explicitly materializing all pairs. These utilities are used only for discovery credit; test reporting retains AUROC and raw/normalized MAE.

B.2 Coalition Estimator

Suppose the current program has B blocks including candidate b. A predecessor coalition is sampled with

$$\nu_b(S) = \frac{1}{B}\binom{B-1}{|S|}^{-1}, \qquad S \subseteq \mathcal{Q} \setminus \{b\}. \tag{15}$$

This gives equal total probability to every coalition size. We draw $M = 8$ coalitions independently with replacement. Empty and full contexts are logged as diagnostics when budget permits, but do not replace random draws unless sampled.

For held-out comparison unit $i \in \mathcal{U}_k$ in fold k,

$$d_i^{(k)}(b; S, \phi) = u_i(S \cup \{b\}, \phi; k) - u_i(S, \phi; k). \tag{16}$$

One context effect and the final credit are

$$\Delta_m(b) = \sum_{k=1}^{K_f} w_k \frac{1}{|\mathcal{U}_k|} \sum_{i \in \mathcal{U}_k} d_i^{(k)}(b; S_m, \phi), \tag{17}$$

where $w_k = |\mathcal{U}_k| / \sum_\ell |\mathcal{U}_\ell|$. The penalized credit is

$$\widehat{c}_b = \frac{1}{M}\sum_{m=1}^{M} \Delta_m(b) - \lambda_c \mathcal{C}(b) - \lambda_\tau \log(1 + \tau_b). \tag{18}$$

The estimator is Shapley-style because it uses the standard predecessor-coalition weighting, but it is local to the current block universe and does not claim task-independent causal attribution.

Let $s_\Delta(b)$ be the sample standard deviation of the M context effects. RelCraft exposes

$$\text{LCB}_\eta^{\text{ctx}}(b) = \widehat{c}_b - \eta s_\Delta(b)/\sqrt{M}. \tag{19}$$

The repeated coalitions reuse held-out units, so this is a context-sensitivity penalty, not a calibrated sampling confidence bound. The action rules are: retain when the lower context score is positive; prune when $\widehat{c}_b + \eta s_\Delta/\sqrt{M} < 0$; replace when the sign changes across contexts; and expand stable, positive path families.

Algorithm 2 Closed adaptive discovery. A credit evaluation compares M base–augmented coalition pairs. A feature proposal is not shortlist-eligible merely because it has credit; only an exactly cross-fitted assembled state enters $\mathcal{V}$.

Require: database D, schema graph G_D, chronological splits, rounds R, coalition count M
Ensure: visited exact pairs $\mathcal{V}$ and audit ledger $\mathcal{L}$

$\mathcal{P} \leftarrow \textsc{TypedPaths}(G_D)$
$h_0 = (\mathcal{Q}_0, \phi_0)$; cross-fit h_0
visited set $\mathcal{V} \leftarrow \{h_0\}$; ledger $\mathcal{L} \leftarrow \varnothing$
for $r = 1, \ldots, R$ **do**
  $(\mathcal{B}_r, \mathcal{M}_r) \leftarrow \textsc{AgentPropose}(\mathcal{P}, h_{r-1}, \mathcal{L})$
  for each feature sketch $s \in \mathcal{B}_r$ **do**
    $q \leftarrow \textsc{Compile}(s)$
    if $\mathsf{Safe}(q) = 1$ **then**
      $(\widehat{c}_q, s_\Delta(q)) \leftarrow \textsc{PairedCredit}(q; \mathcal{Q}_{r-1}, \phi_{r-1}, M, \mathcal{F})$
      store (q, ϕ_{r-1}), contexts, cost, SQL hash, and certificate in $\mathcal{L}$
    else
      store the verifier code; create no model job
    end if
  end for
  for each predictor configuration $\phi \in \mathcal{M}_r$ **do**
    $h_\phi \leftarrow (\mathcal{Q}_{r-1}, \phi)$
    cross-fit h_ϕ; add its measured risk and code length to $\mathcal{V}$
  end for
  $(\mathcal{Q}'_r, \phi'_r) \leftarrow \textsc{AgentEdit}(h_{r-1}, \mathcal{L}, \mathcal{V})$
  $\mathcal{Q}_r \leftarrow \textsc{VerifyAssemble}(\mathcal{Q}'_r)$
  $h_r \leftarrow (\mathcal{Q}_r, \phi'_r)$; cross-fit h_r before adding it to $\mathcal{V}$
  if $\phi'_r \neq \phi_{r-1}$ **then**
    invalidate all ledger entries indexed by $\phi \neq \phi'_r$
    recompute retained-block credit before the next feature-guided edit
  end if
  stop at the first declared budget or after two rounds without positive context score
end for
return $\mathcal{V}, \mathcal{L}$

B.3 Caching and Failures

A cross-fit cache key hashes

$$(D, \mathcal{T}_{\mathrm{tr}}, \mathcal{T}_{\mathrm{dev}}, \mathcal{F}, \mathcal{Q}, \phi, \text{preprocessing}, \text{seed}).$$

Reuse is legal only when every field matches. The base and augmented sides share folds, units, preprocessing, learner settings, and seed. A block rejected statically creates no model job. If a statically valid query later times out or fails dynamically, it receives zero predictive improvement for that context and still incurs its structural and elapsed-time penalties; it is not silently deleted. Every failure remains in the proposal and wall-clock logs.

B.4 Complete Search Procedure

The LLM can act only on measured records. In particular, a proposed predictor configuration is cross-fitted on the current program before it can enter the visited set, and every assembly $h_r = (\mathcal{Q}_r, \phi'_r)$ is cross-fitted as that exact feature–predictor pair. Coalition jobs provide block credit but do not themselves make their temporary coalitions shortlist-eligible. If the predictor changes, all credits indexed by the old predictor are invalidated; any re-crediting is counted as a new member of the next round's credit-request set. This closes the otherwise implicit path from an unevaluated model edit to the Pareto shortlist.

Algorithm 3 One-way protected selection. No protected prediction, loss, rank, or error is returned to discovery.

Require: visited exact pairs $\mathcal{V}$, frozen splits, fixed prior π_0
1: $\mathcal{H} \leftarrow \mathrm{Pareto}(\mathcal{V}; \widehat{\mathcal{R}}_{\mathrm{dev}}, \mathcal{C}, \tau)$; freeze hashes
2: **for** each $h \in \mathcal{H}$ **do**
3: fit its fixed pipeline on $\mathcal{T}_{\mathrm{tr}} \cup \mathcal{T}_{\mathrm{dev}}$
4: evaluate its bounded group loss once on $\mathcal{T}_{\mathrm{sel}}$
5: **end for**
6: choose $h^\star$ by the protected objective in the main paper
7: carry the selected fitted artifact unchanged to $\mathcal{T}_{\mathrm{te}}$
8: **return** frozen SQL–predictor pair $h^\star$

B.5 Fit Accounting

Let $\mathcal{B}_r$ contain every feature-credit request in round r, including re-crediting after a predictor change, and let $\mathcal{M}_r$ contain the predictor proposals. Let R count completed assembled states. Before cache reuse, requested discovery jobs satisfy

$$N_{\mathrm{xf}}^{\mathrm{req}} \le 2M \sum_{r=1}^{R} |\mathcal{B}_r| + \sum_{r=1}^{R} |\mathcal{M}_r| + R + 1. \tag{20}$$

The first term is the paired base/augmented coalition work, the second explicitly evaluates model edits, and $R+1$ covers exact assembled states and initialization. Temporary coalition programs are not placed directly in the visited set, which is why there is no additional per-feature shortlist term. The number reported as "Fits" is

$$N_{\mathrm{xf}}^{\mathrm{uniq}} = |\{\mathrm{hash}(D, \mathcal{T}, \mathcal{F}, \mathcal{Q}, \phi, \mathsf{prep}, \xi)\}| \le N_{\mathrm{xf}}^{\mathrm{req}}.$$

One logical cross-fit job trains $K_f = 5$ fold models. Hence the mean counts in the paper correspond to $5(211.4) = 1{,}057$ atomic fold-model trainings for RelCraft and $5(139.0) = 695$ for RelAgent. Protected candidate evaluations number $|\mathcal{H}|$ and are logged separately from the discovery "Fits" column. Agent-visible validations, logical cross-fit jobs, atomic model fits, database time, and end-to-end wall time are therefore distinct quantities.

C Protected Selection Guarantee

Algorithm 3 summarizes the one-way protected selection step whose guarantee is proved below.

C.1 Fixed Prior and Selection Loss

Fix a task after its schema snapshot and training-only metadata dictionary have been serialized, but before any development or protected outcome is observed. Let $\mathcal{Q}_0$ be the resulting finite dictionary of canonical typed blocks, and let Φ_0 be the finite predictor/preprocessing/seed menu in Appendix D. A candidate has the form $h = (\mathcal{Q}, \phi)$ with $\mathcal{Q} \subseteq \mathcal{Q}_0$ and $\phi \in \Phi_0$; repeated identical blocks are canonicalized away. Denote this finite super-universe by $\mathcal{H}_0$. It contains every pair the search can visit, including pairs later rejected by the verifier or a program-size cap, so using it is conservative.

With

$$\mathcal{C}(h) = \sum_{q \in \mathcal{Q}} \mathcal{C}(q) + \mathcal{C}_\phi(\phi),$$

the fixed normalizer and prior are

$$\begin{aligned} Z_0 &= \sum_{\phi \in \Phi_0} e^{-\gamma \mathcal{C}_\phi(\phi)} \prod_{q \in \mathcal{Q}_0} \left(1 + e^{-\gamma \mathcal{C}(q)}\right), \\ \pi_0(h) &= Z_0^{-1} \exp[-\gamma \mathcal{C}(h)], \qquad \gamma = 1. \end{aligned} \tag{21}$$

The product identity sums exactly over all block subsets, so the normalizer is computable without enumerating $2^{|\mathcal{Q}_0|}$ programs. Restricting to safe, size-capped programs leaves total mass at most

one. The term $\log Z_0$ is common to all candidates but is retained in the reported confidence correction; it is omitted only when an argmin is unaffected. Thus the prior is operational, charges the predictor configuration explicitly, and is independent of both development and protected outcomes.

Protected targets are grouped by entity. For a group G, binary classification loss is its mean Brier loss. Regression loss is

$$\ell_h(G) = \frac{1}{|G|} \sum_{i \in G} \min\{|y_i - \widehat{y}_i|/s_{\mathrm{tr}}, 1\},$$

where s_{tr} is fixed from training labels. Both losses lie in $[0, 1]$. The empirical protected risk is the unweighted mean over n_{sel} entity groups, so repeated targets from one entity do not masquerade as independent observations.

C.2 Theorem and Proof

Theorem 1 (restated)**.** *Condition on the complete adaptive discovery transcript $\mathcal{D}_{\mathrm{disc}}$. Assume the protected entity groups $G_1, \ldots, G_{n_{\mathrm{sel}}}$ are conditionally mutually independent given $\mathcal{D}_{\mathrm{disc}}$, are independent of the adaptive construction of the shortlist, and satisfy $\ell_h(G_i) \in [0, 1]$. Define*

$$\mathcal{R}_{\mathrm{bnd}}(h) = \frac{1}{n_{\mathrm{sel}}} \sum_{i=1}^{n_{\mathrm{sel}}} \mathbb{E}[\ell_h(G_i) \mid \mathcal{D}_{\mathrm{disc}}].$$

Then, for any predeclared $\delta \in (0, 1)$, with probability at least $1 - \delta$, simultaneously for every h in the frozen shortlist $\mathcal{H}$,

$$\mathcal{R}_{\mathrm{bnd}}(h) \le \widehat{\mathcal{R}}_{\mathrm{sel}}(h) + \sqrt{\frac{\log \pi_0(h)^{-1} + \log \delta^{-1}}{2n_{\mathrm{sel}}}}. \tag{22}$$

Proof. Conditioning fixes $\mathcal{H}$, its fitted candidate definitions, and π_0, because the discovery transcript contains no protected outcome. For any fixed h, Hoeffding's inequality for independent, not necessarily identically distributed, bounded group losses gives

$$\Pr\Big[\mathcal{R}_{\mathrm{bnd}}(h) - \widehat{\mathcal{R}}_{\mathrm{sel}}(h) > \epsilon_h\Big] \le \exp(-2n_{\mathrm{sel}}\epsilon_h^2).$$

Set

$$\epsilon_h = \sqrt{\frac{\log \pi_0(h)^{-1} + \log \delta^{-1}}{2n_{\mathrm{sel}}}},$$

so the right-hand side equals $\delta\pi_0(h)$. A union bound over $\mathcal{H}$ yields failure probability

$$\sum_{h \in \mathcal{H}} \delta\pi_0(h) \le \delta \sum_{h \in \mathcal{H}_0} \pi_0(h) \le \delta.$$

Therefore Equation 22 holds simultaneously, including for the data-dependent protected choice $h^\star$. □

The selected candidate minimizes the bound plus the fixed latency penalty. Because the simultaneous event already holds for every shortlisted candidate, adding a deterministic latency term changes the choice but not the validity of the selected candidate's bound. The confidence level δ is a single manifested input fixed before protected evaluation; it is not searched by task or changed after inspecting protected losses.

C.3 Sample Size and Scope

For a desired one-sided slack ϵ, it is sufficient that

$$n_{\mathrm{sel}} \ge \frac{\log \pi_0(h)^{-1} + \log \delta^{-1}}{2\epsilon^2}. \tag{23}$$

The protocol calculates this quantity from the number of protected entity groups, not target rows. For a uniform prior over N candidates, $\log \pi_0(h)^{-1} = \log N$, recovering the familiar logarithmic

Table 8: Coverage of reference results. "Full" denotes a complete result for the applicable suite; "overlap" denotes only tasks reported by the source method. Published aggregates are used for positioning, not paired compute claims.

Family	Method	RB-V1	RB-V2	4DBInfer
Tabular	LightGBM/XGBoost	Full	Full	Full
Automated	DFS+LightGBM	Full	–	Full
Relational	GraphSAGE	Full	Full	Full
Relational	RelGNN/RelGT	Full	–	–
Foundation	RDBLearn	Full	–	Full
Foundation	KumoRFM-v2	Full	Full	Full
LLM-FE	ReFuGe	overlap	–	–
Agentic	RelAgent	Full	Full	Full
Agentic	RelCraft	Full	Full	Full

shortlist dependence. If a task has too few groups, the reported guarantee is simply looser; the method does not reuse development labels or inspect the test period.

The theorem controls the bounded Brier/clipped loss used by selection. It does not imply an AUROC confidence interval, a raw-MAE bound, or independence between temporally connected entities. When entity groups share unmodeled global shocks, Equation 22 should be read as conditional on the declared grouping scheme rather than as a universal time-series guarantee.

C.4 Training and Refit Boundary

Each shortlisted h specifies the program, preprocessing, learner, hyperparameters, and seed. Its protected evaluation fits that complete deterministic pipeline on $\mathcal{T}_{\mathrm{tr}} \cup \mathcal{T}_{\mathrm{dev}}$ without protected labels. The selected fitted artifact is then evaluated on test. This preserves the candidate identity used in Theorem 1. A deployment-only refit that incorporates protected labels may be useful after all reported comparisons are fixed, but it defines a new fitted predictor and is outside both the theorem and the paper's reported test results.

D Experimental and Reproducibility Details

D.1 Benchmark Coverage

The evaluation contains 21 classification and 11 regression tasks. The benchmark training period is $\mathcal{T}_{\mathrm{tr}}$; the earliest 75% of the original validation period is adaptive $\mathcal{T}_{\mathrm{dev}}$, its final 25% is protected $\mathcal{T}_{\mathrm{sel}}$, and the benchmark test period remains $\mathcal{T}_{\mathrm{te}}$. Entity groups never cross a discovery or protected boundary. Within $\mathcal{T}_{\mathrm{dev}}$, the five discovery folds are forward-chaining as described in Appendix B.

A suite mean first averages the five rollout seeds within each task and then weights tasks equally. A missing or timed-out seed retains its predeclared raw-table baseline value rather than being dropped. Incomplete external methods are not inserted into full-coverage average-rank or paired analyses. For regression, task MAE is divided by that task's raw-table LightGBM MAE before aggregation; classification and regression are never combined into one headline number.

D.2 Controlled Resources and Predictor Grid

All controlled methods use the same database snapshot, chronological targets, task-specific split rows, five rollout seeds, predictor menu, DuckDB engine, 900-second query timeout, 60-minute wall-clock cap, and stopping rule. The main controlled comparison uses GPT-5.2; each LLM turn has temperature 1 and at most 8,000 output tokens. A run stops at the first of 60 turns, 30 agent-visible full-program validations, 240 unique discovery cross-fit jobs, 60 minutes, or the method-specific convergence condition. Invalid proposals consume an agent attempt and database time already spent, but a static rejection creates no predictor job.

The finite predictor menu is:

Table 9: Search and selection coefficients. The single tuple is fixed across tasks, suites, and LLM backbones; there is no hidden per-benchmark coefficient search range.

Quantity	Value	Fixed role
λ_c	10^{-3}	normalized structural penalty
λ_τ	10^{-3}	measured materialization penalty
λ_e	10^{-3}	predicted acquisition-time penalty
η	1	one context standard-error penalty
κ	10^{-2}	under-tested path-family bonus
γ	1	exponential prior slope in code units

- **LightGBM**: GBDT, DART, or GOSS; leaves $\{31, 63\}$; depth $\{-1, 8\}$; learning rate $\{0.03, 0.10\}$; estimators $\{300, 600\}$.
- **XGBoost**: tree or DART booster; depth $\{4, 8\}$; learning rate $\{0.03, 0.10\}$; estimators $\{300, 600\}$; subsample and column subsample fixed at 0.8.
- **Random forest**: trees $\{300, 600\}$; depth $\{\text{unbounded}, 12\}$; minimum leaf size $\{1, 5\}$.
- **CatBoost**: depth $\{6, 8\}$; learning rate $\{0.03, 0.10\}$; iterations $\{300, 600\}$.

Classification optimizes log loss during fitting and reports AUROC. Regression permits squared, absolute, or Huber training objectives when supported. The target transform is identity or `log1p` for nonnegative targets; transform eligibility is fixed from training metadata. Early stopping uses only the training side of the current fold. Category vocabularies, imputation values, normalization scales, and target-transform constants are recomputed inside every training fold.

The menu contains 80 learner-structure settings before eligible regression objectives and target transforms are applied: 48 LightGBM, 16 XGBoost, 8 random-forest, and 8 CatBoost settings. A predictor proposal names one complete configuration; it cannot add code or alter an unlisted option. Unlisted library options remain fixed at the environment defaults recorded in the run manifest. Thus a configuration enters the visited set only through the explicit cross-fit in Algorithm 2.

Search coefficients are fixed across tasks:

$$(\lambda_c, \lambda_\tau, \lambda_e, \eta, \kappa, \gamma) = (10^{-3}, 10^{-3}, 10^{-3}, 1, 10^{-2}, 1).$$

For search penalties, structural length is normalized as $\widetilde{\mathcal{C}} = \mathcal{C}/\mathcal{C}_{\max}$ using the maximum of the frozen task language; the protected prior uses the unnormalized code in Equation 21. For raw materialization time τ_{sec}, the normalized cost term is

$$r_\tau(q) = \frac{\log(1 + \min\{\tau_{\text{sec}}(q), 900\})}{\log(901)} \in [0, 1],$$

so the main paper's $\log(1 + \tau)$ cost is implemented by r_τ (not by applying a second logarithm). A timeout receives $r_\tau = 1$. The acquisition term uses a prediction on this same scale. The roles are summarized in Table 9.

D.3 Metric Definitions

For classification $V = \text{AUROC}$; for regression $V = 1 - \text{NMAE}$, with both represented on the 0–1 utility scale. Let h_{dev} be the development choice and $h^\star$ the protected choice. The reported gaps are metric *points*:

$$\text{Gap}_{\text{dev}} = 100|V_{\text{dev}}(h_{\text{dev}}) - V_{\text{te}}(h_{\text{dev}})|, \tag{24}$$

$$\text{Gap}_{\text{sel}} = 100|V_{\text{sel}}(h^\star) - V_{\text{te}}(h^\star)|. \tag{25}$$

These are calibration diagnostics, not oracle regret. For baselines that do not use protected selection, the protected–test value is computed only after the baseline candidate has been frozen; it is never fed back to that baseline.

For the common budget $N = 240$ unique discovery cross-fit jobs, let h_j be the best exactly evaluated valid state available after job j. Then

$$\mathrm{AUTC}_N = \frac{1}{N} \sum_{j=1}^{N} \max_{0 \le k \le j} V_{\mathrm{dev}}(h_k).$$

The running maximum preserves an earlier valid program after a failed or worse proposal. If a method terminates before N, its last running best is carried forward through job 240, so a shorter trajectory does not receive a smaller denominator.

Agent-visible validations use a separate index. Let g_v be the full program evaluated at validation v, let J be the final number of such validations, and define the final running-best gain

$$G_J^{\mathrm{val}} = \max_{0 \le r \le J} V_{\mathrm{dev}}(g_r) - V_{\mathrm{dev}}(g_0).$$

The "Val." entry is

$$J_{95} = \min \left\{ v : \max_{0 \le r \le v} V_{\mathrm{dev}}(g_r) \ge V_{\mathrm{dev}}(g_0) + 0.95 G_J^{\mathrm{val}} \right\}. \tag{26}$$

If $G_J^{\mathrm{val}} = 0$, $J_{95} = 0$ by convention. Thus AUTC is an absolute fit-budget trajectory, whereas J_{95} counts agent decisions required to recover 95% of the eventual running-best *gain*, not 95% of absolute AUROC.

For the backbone table, with T the final turn (or earlier stopping turn),

$$\mathrm{Rec@10} = 100 \, \frac{\max_{r \le 10} V_{\mathrm{dev}}(g_r) - V_{\mathrm{dev}}(g_0)}{\max_{r \le T} V_{\mathrm{dev}}(g_r) - V_{\mathrm{dev}}(g_0)},$$

with the zero-gain convention above. Cross-rollout deviation is $\sigma_{\mathrm{roll}} = 100 \, \mathrm{sd}_{s=1}^{5}[V_{\mathrm{te}}(h_s^\star)]$, computed per task before equal task weighting.

Let N_{prop} be the number of agent-proposed feature sketches and let N_{fail} count those that fail verification, dry run, or execution. SQL failure is

$$\mathrm{SQLFail} = 100 \, N_{\mathrm{fail}} / N_{\mathrm{prop}}.$$

It is computed per task–seed before the same seed-within-task and equal-task aggregation. "Fits" is $N_{\mathrm{xf}}^{\mathrm{uniq}}$ from Appendix B, not atomic fold models. Blocks counts canonical deployed feature blocks. Latency includes selected SQL materialization, preprocessing, and prediction after one warm-up, is normalized to seconds per 10K targets, and excludes discovery, LLM, and coalition work.

D.4 Uncertainty and Paired Analysis

The mean±standard deviation in the main predictive table is computed from the five rollout-level suite means. With $n = 5$ and $t_{0.975,4} = 2.776$, a seed-level interval is

$$\bar{x} \pm t_{0.975,4} \frac{s}{\sqrt{5}}.$$

Table 10 lists the resulting intervals, and Table 11 decomposes the common-harness RelCraft–RelAgent comparison into absolute and relative changes.

The benchmark-positioning deltas are consistent across all five reported suite metrics: relative to the published RelAgent aggregate, classification AUROC is higher by 0.96, 0.82, and 0.88 points on RelBenchV1, RelBenchV2, and 4DBInfer; NMAE is lower by 0.024 (2.9%) and 0.020 (3.6%) on the two regression suites. These are descriptive comparisons because the external row is not a matched rerun. RelCraft is also 0.26 AUROC points below KumoRFM-v2 on RelBenchV1, so the evidence supports broad competitiveness rather than uniform dominance.

Table 10: Intervals derived from the reported five-seed means and standard deviations. They measure rollout variation and are not paired tests against published aggregates.

Metric	Suite	Mean	Std.	95% seed CI
AUROC	RelBenchV1	79.34	0.22	[79.07, 79.61]
AUROC	RelBenchV2	88.14	0.19	[87.90, 88.38]
AUROC	4DBInfer	82.26	0.25	[81.95, 82.57]
NMAE	RelBenchV1	0.793	0.006	[0.786, 0.800]
NMAE	RelBenchV2	0.528	0.004	[0.523, 0.533]

Table 11: Effect-size decomposition of the main paper's common-harness RelCraft–RelAgent comparison. Relative changes use RelAgent as the denominator. These values are algebraic summaries of the reported means, not additional trials or substitutes for task-level paired inference. The fit row makes the central trade-off explicit: RelCraft improves decisions, reliability, and deployment while spending more offline model work.

Controlled metric	RelAgent	RelCraft	Change (relative)
AUTC	0.781	0.814	$+0.033$ ($+4.2\%$)
Val. to 95% gain	27.8	21.6	-6.2 (-22.3%)
Logical cross-fit jobs	139.0	211.4	$+72.4$ ($+52.1\%$)
SQL failure (%)	6.8	1.9	-4.9 pp (-72.1%)
Dev/Test gap (points)	1.54	1.02	-0.52 (-33.8%)
Sel/Test gap (points)	1.31	0.56	-0.75 (-57.3%)
σ_{roll} (points)	0.91	0.51	-0.40 (-44.0%)
Selected blocks	46.8	31.6	-15.2 (-32.5%)
Latency (s/10K)	7.4	5.8	-1.6 (-21.6%)

For a controlled comparison, the inferential unit is a task. We first average seeds within task, form paired task differences, and then use a task bootstrap that resamples whole tasks with all their seeds attached. A two-sided randomization test flips the sign of each task difference; when several metrics support one claim family, Holm correction is applied within that family. Rows, class pairs, and repeated targets are never treated as independent replicates for a system-level claim. Because the external rows in the predictive table are published aggregates rather than matched reruns, no paired significance statement is made against them.

D.5 Backbone and Early-Budget Robustness

For the backbone comparison in Table 6, the predictor menu, database resources, split rows, seeds, and stopping budgets are held fixed across backbones. RelCraft improves AUROC by 0.92, 1.02, and 1.67 points for GPT-5.2, DeepSeek-V4, and Qwen3-32B, respectively, and improves Rec@10 by 7.8, 8.6, and 9.4 points. Averaged across the three backbones, the gains are 1.21 AUROC and 8.6 Rec@10 points. RelCraft's AUROC range is 0.71 points compared with 1.46 for RelAgent. This is descriptive robustness evidence under fixed resources, not three independent benchmark replications.

D.6 Mechanism-Level Interpretation

The ablation table in the main paper separates the mechanisms rather than asking one headline metric to support every claim. Removing coalition credit reduces offline jobs from 211.4 to 139.0 but loses 0.65 RelBenchV1 AUROC points, enlarges the protected–test gap from 0.56 to 0.91, and retains 10.5 additional blocks. Leave-one-out credit only partly recovers these changes (78.91 AUROC, 0.83 gap, and 38.7 blocks), which is consistent with the need for multiple interaction contexts rather than simply more evaluations.

Removing protected selection produces the largest ablation gap, 1.41 versus 0.56, while changing AUROC by only -0.29 points; this is the designated evidence that the protected boundary concerns selection calibration rather than a claim of large mean-accuracy gain. Removing the complexity prior changes AUROC by only -0.03 points but expands the deployed program from 31.6 to 49.8

blocks (57.6%). Removing execution cost slightly raises AUROC by 0.02 points while expanding the program to 47.2 blocks (49.4%). These two rows show an explicit accuracy–deployability trade-off and prevent the 31.6-block result from being attributed only to weaker prediction.

D.7 Controlled Probe Construction

Triangle is a two-hop task whose label depends on a typed triangular join motif. **XOR** supplies two binary relational aggregates that are individually uninformative but jointly determine the label by parity. A forward, single-context method cannot retain either block from standalone credit. **Leak/2K** is a frozen bank of 2,000 adversarial sketches containing an explicit post-cutoff predicate on an event-bearing edge. **Sign-F1** compares the sign of estimated credit with the exact size-balanced coalition mean obtained by enumerating all coalitions in the small probe. Probe generators, seeds, and sketch banks are fixed before method evaluation.

The probe values in the main paper give three direct contrasts. Typed scalar generation raises Triangle AUROC from 0.842 to 0.996 and reduces explicit future-edge admission from 187/2,000 (9.35%) to 0/2,000. Typing alone leaves XOR at chance (0.500), whereas RelCraft reaches 1.000; this rules out the typed grammar as the explanation for that interaction result. Finally, multi-context credit raises Sign-F1 from 0.731 for typed leave-one-out credit to 0.914. These are controlled stress tests for expressivity, explicit cutoff enforcement, and credit signs; they are not estimates of natural-task prevalence.

D.8 Claim–Evidence Map

Table 12 assigns one designated evidence track to each claim.

Table 12: One designated evidence track per claim. This separation avoids using benchmark averages as evidence for safety, or a synthetic probe as evidence for natural-task accuracy.

Claim	Designated evidence, supported reading, and boundary
Point-in-time control	SQL failure, Leak/2K, and safety ablations support rejection of declared invalid/future edges; they do not prove upstream lineage or absence of target proxies.
Interaction-aware credit	Coalition/LOO ablations, XOR, and Sign-F1 support a better context-sensitive search signal, not causal attribution.
Protected selection	Sel/Test gap and the w/o-protected ablation support better frozen-candidate calibration, not an AUROC theorem.
Joint feature–model search	Algorithms 2–3 and the fit ledger show that only measured exact pairs are shortlisted, at more offline fits than RelAgent.
Deployability	Block count, latency, and cost ablations support a smaller, faster selected program while preserving the accuracy–cost trade-off.
Predictive positioning	Complete-suite aggregates support consistent descriptive suite gains; external rows are not paired reruns.
Backbone robustness	The three-backbone controlled table supports persistence under a fixed action space and budget; backbones are not new tasks.

D.9 Failure Audit and Run Manifest

Every unsuccessful proposal receives exactly one primary cause from the first failing stage: unknown schema reference, invalid feature type, missing/non-strict cutoff, row-expansion/resource cap, or execution timeout. This first-failure rule makes cause counts disjoint. We record both proposal and rejection rates: a verifier rejection consumes an agent-visible attempt but no predictor fit, whereas a runtime failure also consumes database time. For accepted programs, a fixed zero-label audit checks duplicate target rows, post-cutoff source rows, split-dependent output columns, and feature-matrix hashes.

Every run manifest stores:

- database snapshot, task, split-row hashes, and target cutoff policy;

- LLM backbone, sampling settings, prompt hash, and turn budget;
- sketch, normalized SQL, SQL hash, verifier certificate, and error code;
- coalition, folds, predictor configuration, preprocessing hash, and seed;
- out-of-fold predictions, metric implementation version, and cache status;
- logical jobs, atomic fold fits, SQL time, training time, peak memory, and end-to-end time;
- coefficient tuple, prior normalizer, confidence input δ, and protected entity-group count;
- frozen shortlist hash, protected-selection record, and final fitted-artifact hash.

Table generation reads only these manifests and stored predictions. Test labels remain sealed until all rollouts, candidate sets, aggregation code, and analysis choices are fixed. Reproduction rebuilds features from sketches, reruns verifier corner cases, recomputes metrics from predictions, and replays protected selection without an LLM.

E LIMITATIONS

RelCraft establishes reliability only within its declared typed program language. The verifier checks explicit schema edges, target-relative cutoffs, split isolation, row alignment, and resource bounds, but it cannot recover provenance omitted from an upstream database snapshot or determine whether a semantically permissible field is a target proxy. Coalition credit is a search heuristic, not a task-independent causal estimand, and its $M = 8$ contexts introduce Monte Carlo variation. The method also spends more training compute than RelAgent even though it reduces agent-visible decisions and yields a better fit trajectory. Finally, Theorem 1 applies only to the bounded group loss used for protected selection under independent entity groups; it does not guarantee AUROC or raw MAE.

F BROADER IMPACTS

Reliable point-in-time feature construction can reduce silent temporal leakage and make relational prediction pipelines easier to inspect and reproduce. However, automated feature search may also amplify biases, privacy risks, or proxy discrimination already present in a relational database. Deployment therefore still requires data-governance review, access control, provenance auditing, and task-specific fairness evaluation. RelCraft's verifier should be treated as one technical safeguard rather than a substitute for those controls.