A Ablation Studies

We present several ablation studies. First, we show the impact of different joinability configurations. We then show how MATILDA can cover 100% of AMIE's rules, even those that connect non-joinable attributes.

Joinability. We investigate three joinability configurations:
FK Joinability: Serving as the default in our main experiments, joinability is determined exclusively by foreign key relationships. This method leverages schema-defined constraints to ensure that only semantically meaningful joins are evaluated, thereby reducing the search space.
Rule-Based Joinability: Joinability is established using pruning rules focused on attribute overlap and data type compatibility. Two attributes are joinable if:

(1) *Overlap*: The values of the attributes exhibit a Jaccard similarity of at least **50%**, or they overlap at least **70%** in both directions.
(2) *Type Compatibility*: The attributes are declared to be of the same domain (type). Numerical values are not considered joinable, to avoid accidental overlapping, such as between an ID and a person's age.

Full Joinability: Every attribute is considered joinable with all other attributes. Full joinability removes all of MATILDA's pruning efforts and is usually not desirable as it generates or tests rules such as $Lineage(parent = x_0) \Rightarrow Residence(state = x_0)$. We present it here just as an ablation.

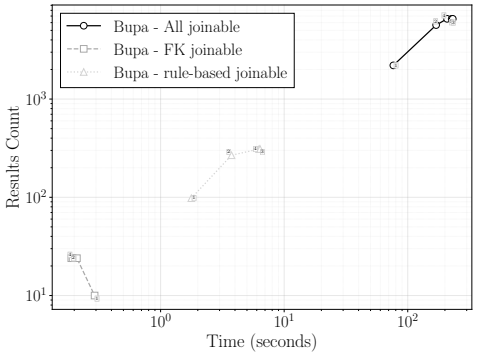

Fig. 4. MATILDA's performance under different joinability settings, with recursion depth varied from 1 to 3 (log-log scale).

For our experiments, we chose the Bupa database, as it was the only database that had more than one table and could be mined with full joinability in less than a day. We evaluate each joinability setting recursion depths of 1, 2, and 3. Figure 4 shows the results. The FK-only joinability setting remains relatively unaffected by recursion depth, with negligible time variations (approximately 0.01 seconds) between the lower and higher levels of recursion. Conversely, both the fully joinable and the rule-based setting exhibit a notable increase in computation time and result set size as recursion depth increases. This observation confirms our choice of constraining joinability to foreign keys.

Coverage of AMIE Rules. The rules that AMIE mines on our databases usually carry only limited insight, because they often concern a single row in a table and join attributes that are not semantically joinable, as discussed in Section 6.3. However, if the user wishes to find these rules also with MATILDA, this is possible: MATILDA has to be run with `joinable=ALL` (which makes all columns joinable), `recursion=2` (which allows the same table to appear at most twice in a TGD),

	AMIE 3		MATILDA		
Database	**Results**	**Time**	**Results**	**Time**	**Cover.**
CSS	2	53.09s	202	38m52s	100%
CraftBeer	1	2.21s	62	1.56s	100%
Dallas	3	1.34s	96	4.83s	100%
SFScores	8	12.18s	146	4m28s	100%
WebkP	2	3.66s	26	9.3s	100%

Table 6. Comparison of AMIE 3 with MATILDA using the parameters `joinability=True`, `recursion=2`, `max_table=2`. **Coverage** indicates the percentage of AMIE's rules that are covered by MATILDA.

`max_table=2` (which mines at most two tables per TGD), and `max_var=1` (which allows at most one variable per TGD). We identified 5 databases where both AMIE and MATILDA (with this configuration) mine rules, and show the results in Table 6. MATILDA is up to 40 times slower than AMIE. This is because, in the absence of joinability constraints, the constraint graph is exponentially large, and, unlike AMIE, MATILDA has to explore it in full in order to find full-fledged TGDs. In return, MATILDA mines not just the (often spurious) rules of AMIE, but up to 100 times more patterns. For normal applications, we recommend the default setting of MATILDA, which finds only those TGDs that join joinable attributes.

More ablation studies, which study the impact of sampling rows and columns, are in Appendix B.

Database	**Rule**	**Interpretation**
Dunur	$\forall x_0$: husband(name1=x_0) $\Rightarrow \exists z_0$: brother(name2=z_0) $\wedge$ uncle(name2=z_0, name1=x_0)	Every husband x_0 has a brother z_0 who is also x_0's uncle, reflecting a complex familial relationship.
Mutagenesis	$\forall x_0$: molecule(molecule_id=x_0) $\Rightarrow \exists z_0$: atom(atom_id=z_0, molecule_id=x_0) $\wedge$ bond(atom2_id=z_0)	Every molecule contains at least one atom that participates in a bond, a meaningful chemical structure constraint.
Student Loan	$\forall x_0$: disabled(name=x_0) $\Rightarrow \exists z_0$: bool(name=z_0) $\wedge$ no_payment_due(bool=z_0, name=x_0)	For every disabled individual x_0, there exists a boolean indicator ensuring that no payment is due.
UW	$\forall x_0$: course(course_id=x_0) $\Rightarrow \exists z_0$: advisedBy(p_id_dummy=z_0) $\wedge$ taughtBy(p_id=z_0, course_id=x_0)	Every course x_0 has at least one person z_0 who both advises and teaches it, indicating a structural pattern in course management.
Same-gen	$\forall x_0$: same_gen(name2=x_0) $\Rightarrow \exists z_0$: parent(name1=x_0, name2=z_0) $\wedge$ person(name=z_0)	Entities marked as same generation have a parent-child relationship, revealing generational or hierarchical structures.
Biodegradability	$\forall x_0$: atom(molecule_id=x_0) $\Rightarrow$ molecule(molecule_id=x_0)	Every atom is linked to a molecule.
Country	$\forall x_0$: Country(Code=x_0) $\Rightarrow$ City(CountryCode=x_0)	Every Country has a related city

Table 7. Examples of TGDs that only MATILDA mines.

B Sampling Rows and Columns

Sampling rows. The complexity of MATILDA depends mainly on the number of attributes, and less on the number of rows. To demonstrate this experimentally, we down-sampled the rows of the Bupa dataset, and measured MATILDA's runtime at different sampling rates. Figure 6 shows that a sampling results in a linear decline in the number of results. However, this effect almost vanishes in comparison to the effect of the recursion depths: The log-log scale clearly shows that higher recursion depths (marked as increasing labels along the curves) contribute significantly to the results count, irrespective of the sampling rates. At lower sampling rates (e.g., 10%), the time taken to compute results remains shorter, but fewer rules are discovered. Conversely, higher sampling rates (e.g., 90%) lead to increased computational overhead due to the larger search space, but they allow for a more exhaustive rule discovery process.

This relationship highlights the tradeoff between computational efficiency and the comprehensiveness of rule mining when adjusting sampling parameters and recursion depths. The figure also suggests that the impact of recursion depth becomes more pronounced as the row sampling rate increases, showing the exponential nature of the search space expansion in rule mining tasks.

Sampling columns. Figure 7 illustrates the effects of reducing the number of columns on the mining process. Unlike row sampling, reducing the number of available columns significantly impacts performance, both in terms of computational efficiency and the number of extracted rules. Specifically, while row reductions can decrease the number of discovered rules by a factor of approximately 10^2, column reductions can lower results by an order of magnitude of 10^3. This suggests that column selection directly influences rule discovery, as reducing the feature space limits the number of possible dependencies that can be inferred.

Furthermore, as recursion depth increases, the impact of column pruning becomes even more evident. A reduced attribute space leads to a significant decline in the number of generated rules. However, due to the randomness in column trimming, certain edge effects may occur, leading to cases where different levels of pruning (e.g., 10% vs. 40%) yield similar results. This irregularity arises from the specific columns removed in each iteration, as some attributes may be more pivotal in rule formation than others.

B.1 Analysis of Joinable Attributes and Runtime Performance

The number of joinable attributes in a database significantly influences the search space and runtime of pattern mining algorithms. Figure 5 shows the relationship between the number of joinable attribute pairs and MATILDA's execution time. We observe that MATILDA scales sublinearly with the number of joinable constraints, which validates the efficiency of our constraint-based search strategy.

Databases with more joinable attributes generally require longer execution times, but the relationship is not strictly linear due to: (1) the pruning effectiveness of our constraint propagation, (2) the distribution of data values affecting pattern frequency, and (3) the complexity of the schema structure beyond simple attribute counts.

C Complex TGDs

When the joinability is set to Foreign Key Joinability, MATILDA finds only few complex TGDs. When we relax the joinability to Rule-based (Section A), more TGDs are unearthed, and we show some examples in Table 7.

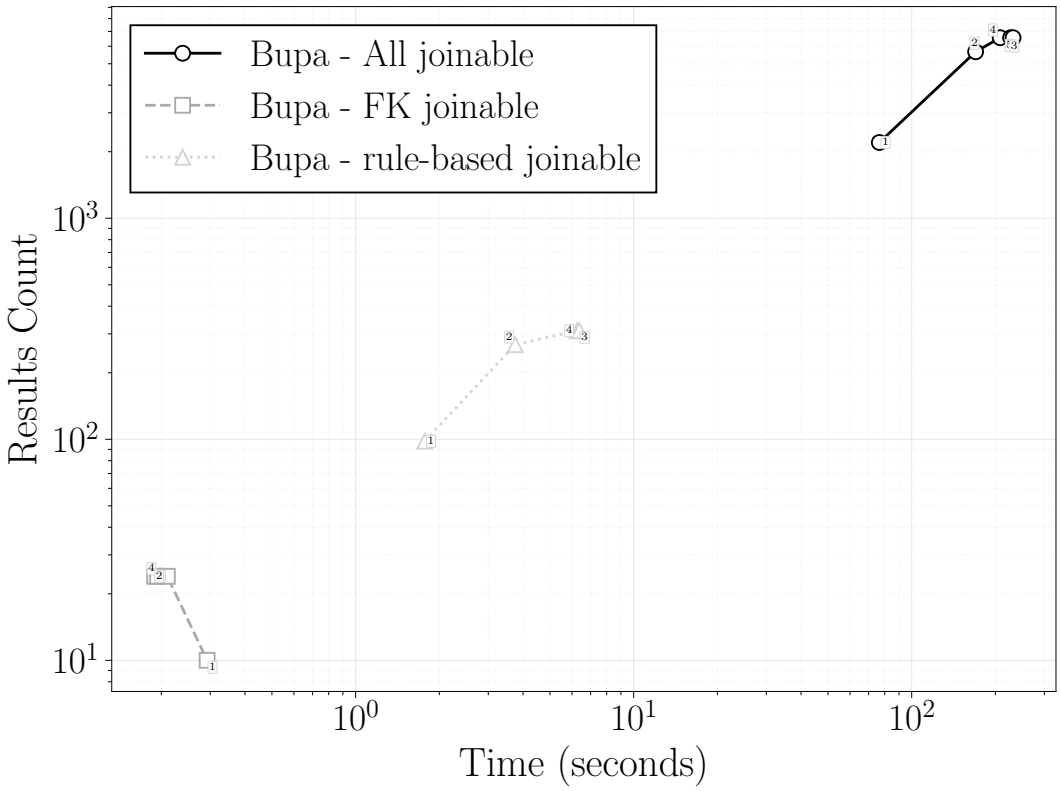

Fig. 5. Correlation between number of joinable attribute pairs and MATILDA runtime. The relationship shows sublinear scaling, demonstrating the efficiency of our constraint-based approach.

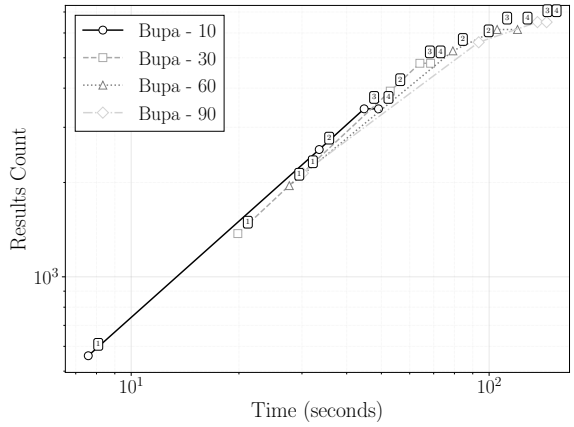

Fig. 6. Performance with varying recursion depth (1,2,3,4) and row sampling rates (log-log-scale).

D Ablation Study: Impact of Database Complexity on MATILDA Performance

We present a comparative ablation analysis of MATILDA's performance across two critical dimensions: **execution time** and **rule discovery count**. This study reveals distinct patterns and correlations for each performance aspect, providing insights into the algorithm's computational behavior and effectiveness.

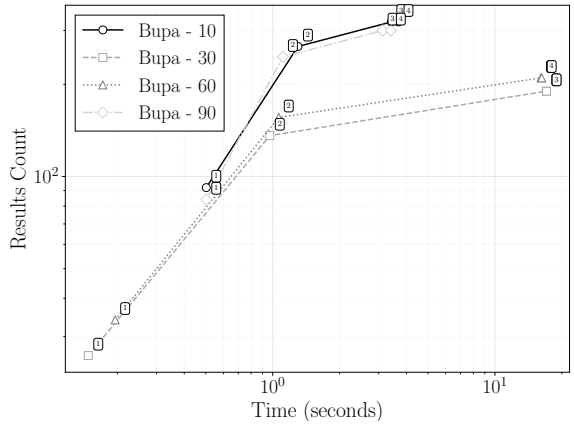

Fig. 7. Performance with varying recursion depth (1,2,3,4) and column sampling rates (log-log-scale).

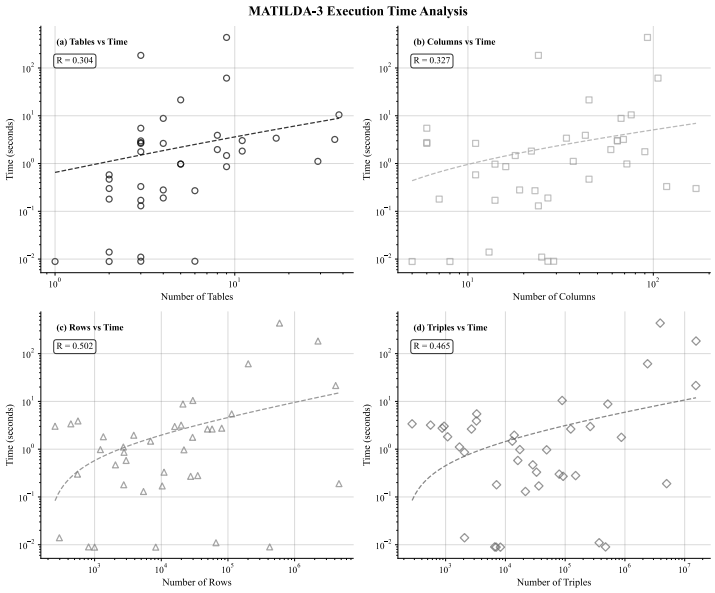

Fig. 8. Correlation between number of columns and number of discovered rules.

Fig. 9. Correlation between number of tables and number of discovered rules.

Methodology. We evaluated MATILDA with two complementary regressions: a *time analysis* linking execution time to schema and data properties, and a *rule-count analysis* relating the number of discovered rules to the same properties. Five database metrics were considered throughout—table count, column count, row count, triple count, and the foreign-key–to-column ratio.

Key findings. Execution time rises chiefly with data volume, showing its strongest correlations with rows ($R = 0.50$) and triples ($R = 0.47$), a modest correlation with columns ($R = 0.33$), and a negligible link to tables. Rule discovery, by contrast, is driven by schema complexity: tables correlate most strongly ($R = 0.56$) and columns moderately ($R = 0.34$), whereas dense foreign-key usage ($R = -0.33$) and larger data volumes ($R \approx -0.25$ to -0.30) suppress discovery. These opposing

patterns reveal that MATILDA's implicit-relationship heuristics thrive on rich schemas but are impeded by sheer data size.

Practical implications. For time-critical scenarios, practitioners should limit rows (and therefore triple counts); for maximal rule extraction, they should favour table-rich, row-lean designs, accepting some runtime overhead. Balanced deployments pair moderate schema complexity with controlled data volumes. Developers are encouraged to optimise row-level processing and reinforce large-scale pattern recognition, while researchers should model runtime and rule yield separately.

Conclusion. MATILDA exhibits a dual performance nature: execution time is governed by data volume, whereas rule discovery hinges on schema structure, necessitating distinct optimisation strategies for each objective.