



KAIST Hacking Lab

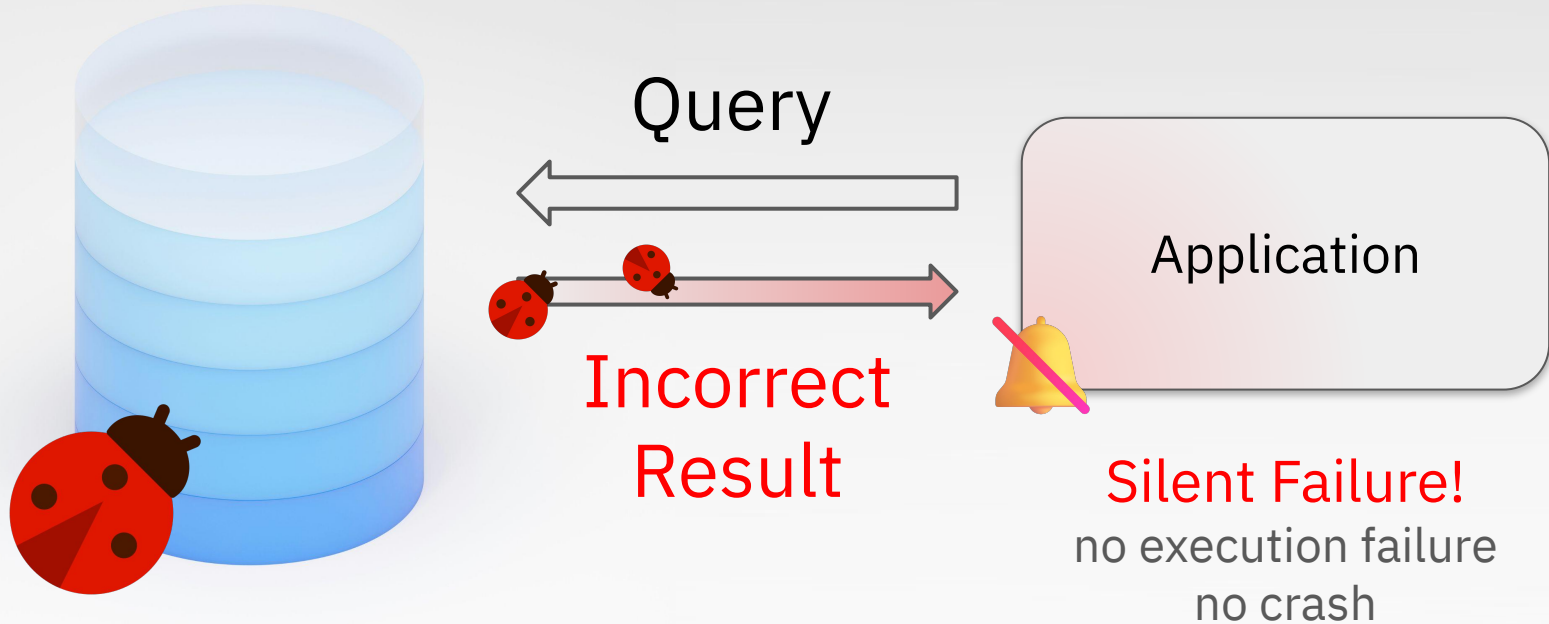
QueryHouse

Cross-DBMS Differential Testing with LLM and Query Transpilation

Seoyeon Oh¹, Juhyun Song¹, Jaemin Park², Kwanghee Lee³, Minae Hyeon⁴,
Minji Kim⁵, Sehyuk Ka⁶, Gwangwun Jung⁷, Brian Pak⁷ and Insu Yun¹

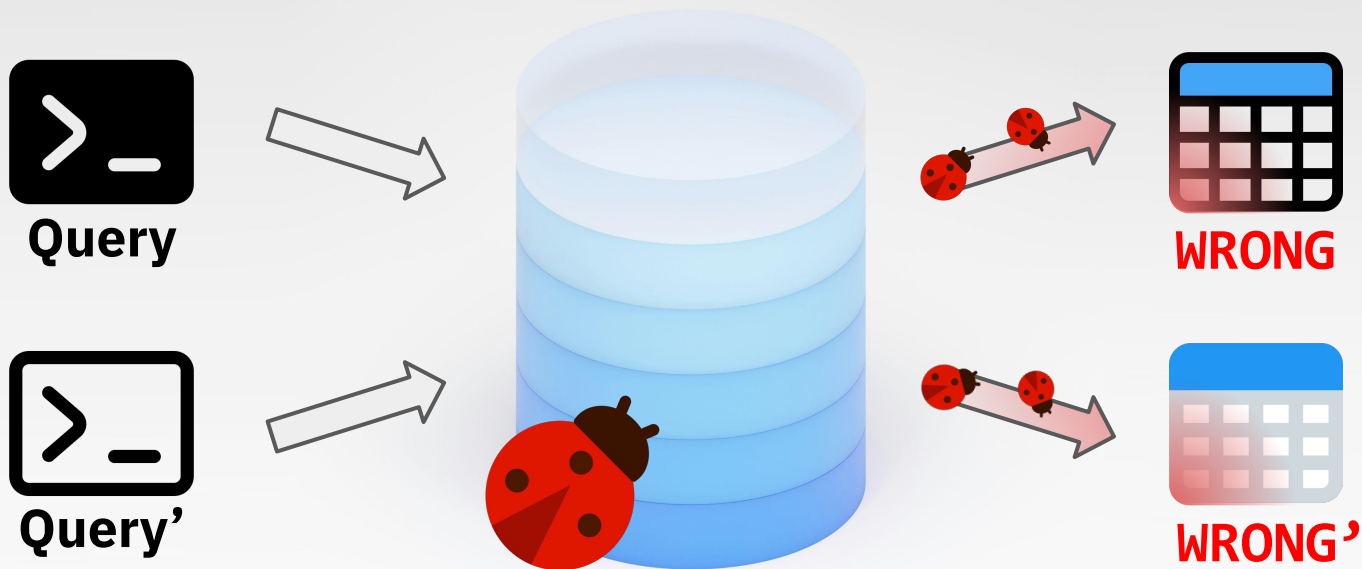
DBMS Logic Bugs

- DBMS logic bugs can ***silently*** affect the application on top of it.



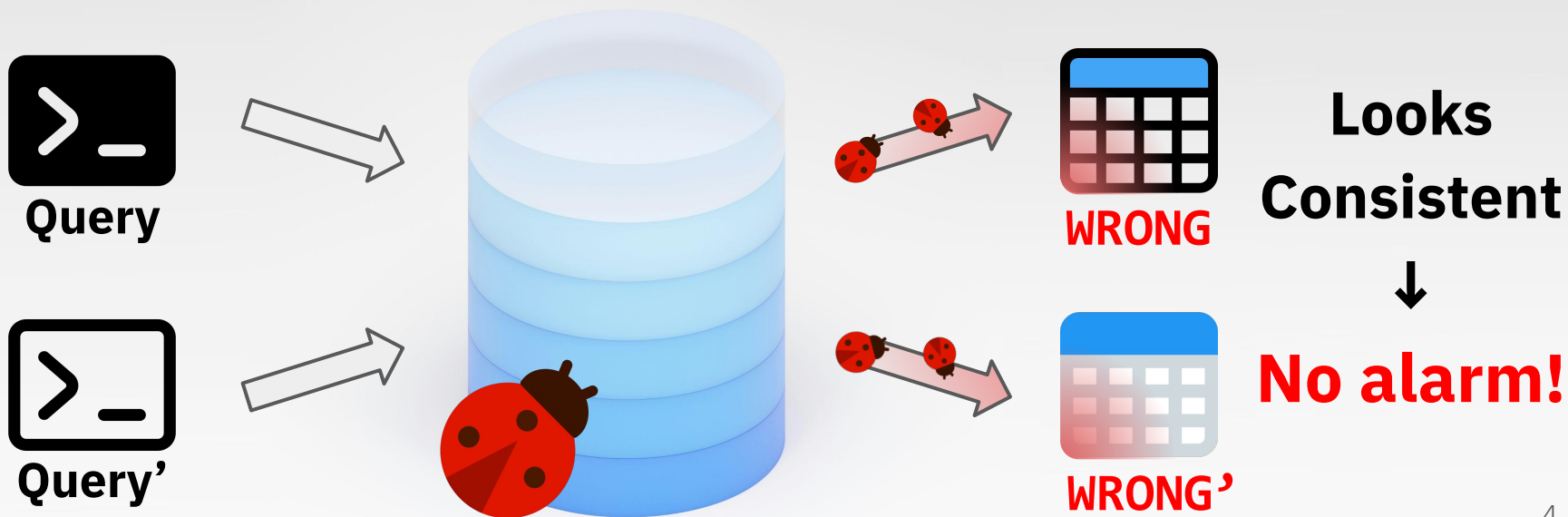
The Blind Spot: Sympathetic Failures

- Existing works focus on **metamorphic testing**.
- It tests **within a single DBMS** using carefully designed query transformation.



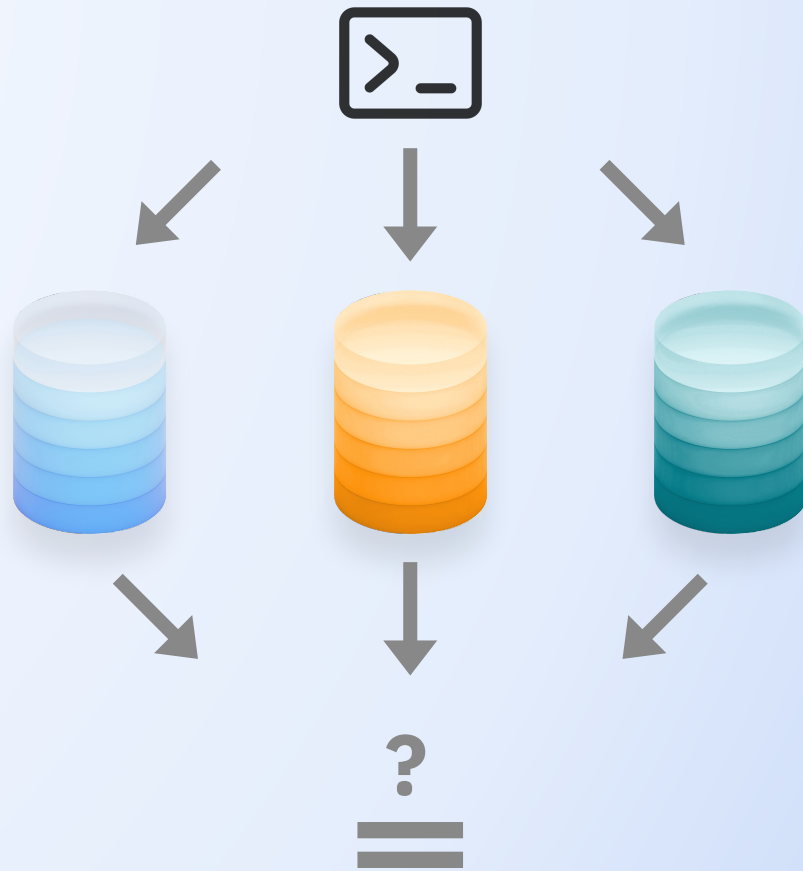
The Blind Spot: Sympathetic Failures

- Existing works focus on **metamorphic testing**.
- It tests **within a single DBMS** using carefully designed query transformation.



Differential Oracle

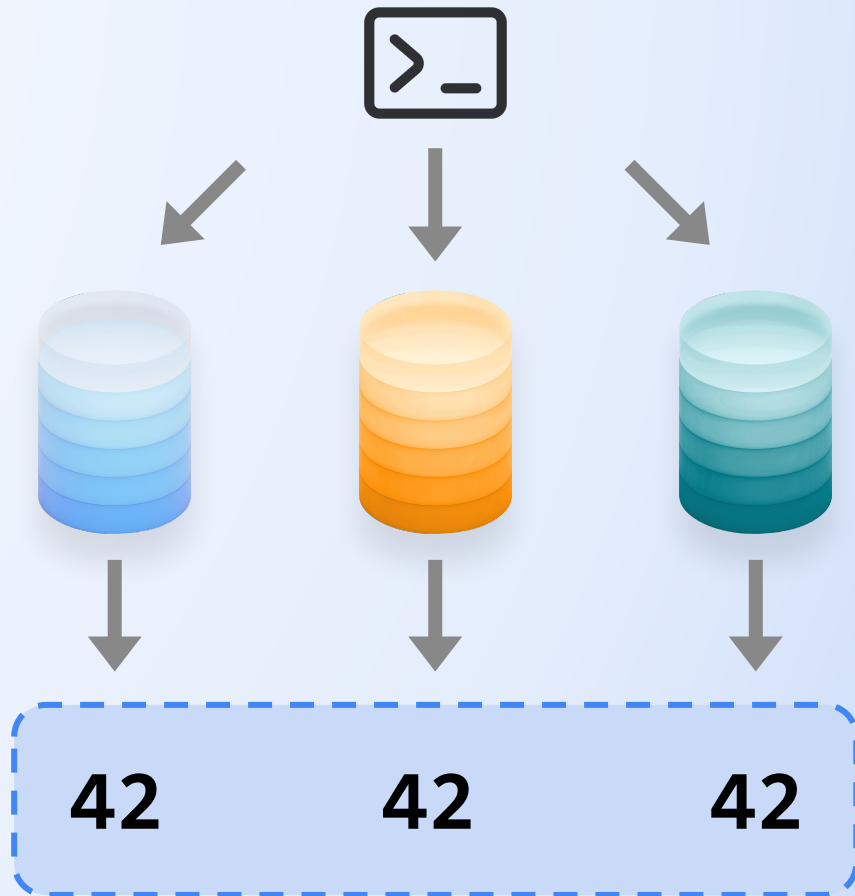
- Use **other DBMSs** as the oracle.



Differential Oracle

- Use **other DBMSs** as the oracle.

✓ **Correct!**

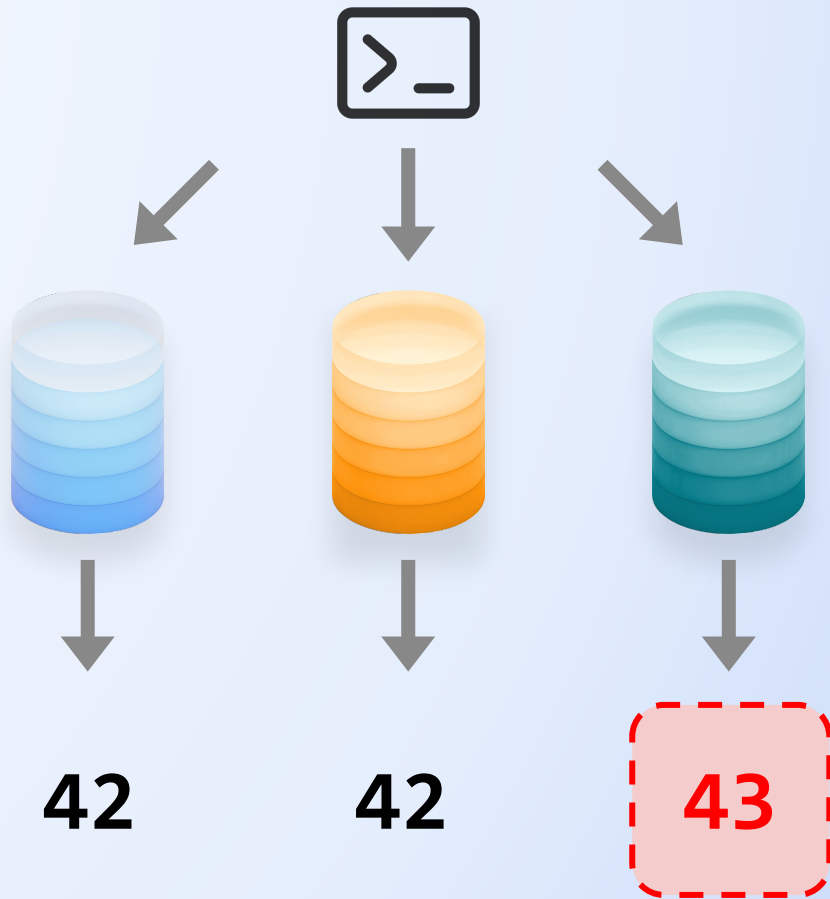


Differential Oracle

- Use **other DBMSs** as the oracle.
 - Different results indicate potential bugs.



MISMATCH!



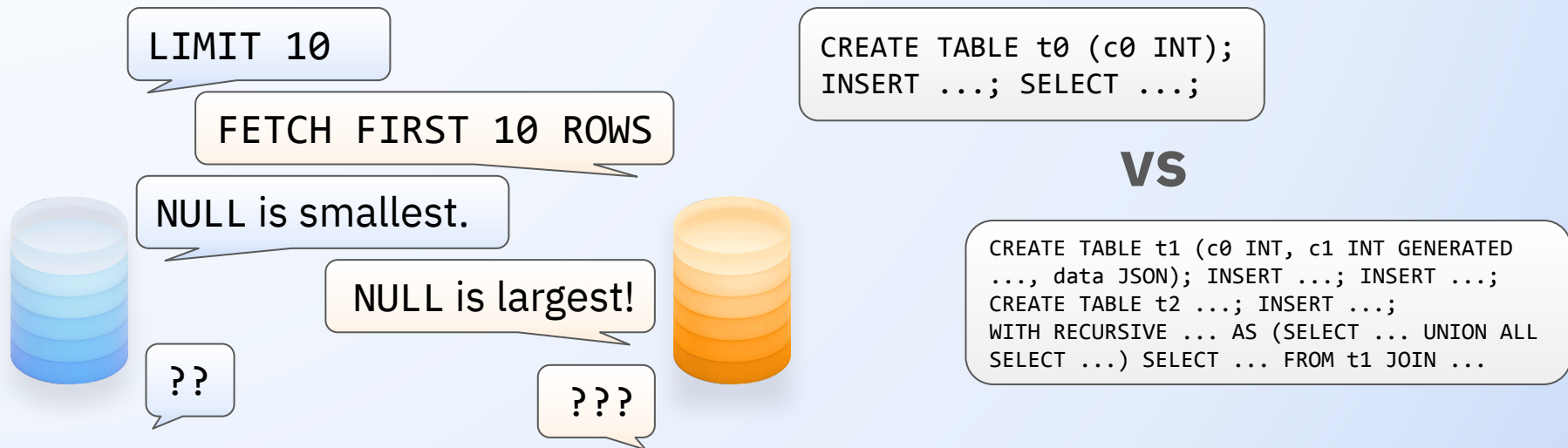
Challenges in Cross-DBMS Testing

- **SQL discrepancy**

- DBMSs speak their own different dialects, diverging in syntax and semantics.

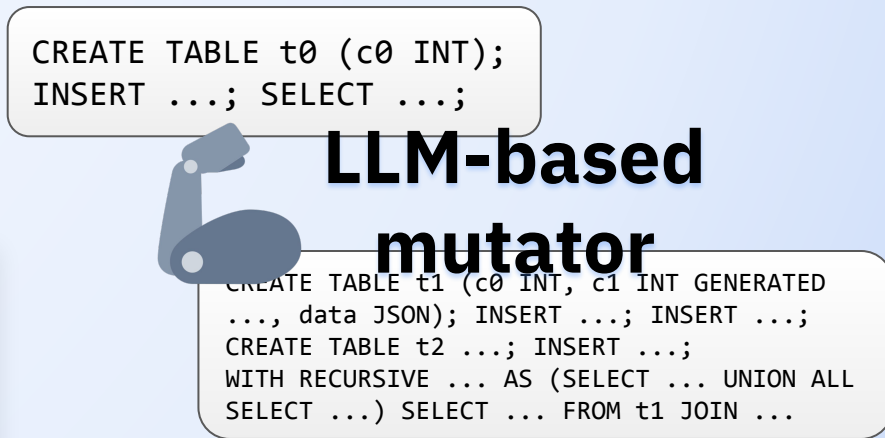
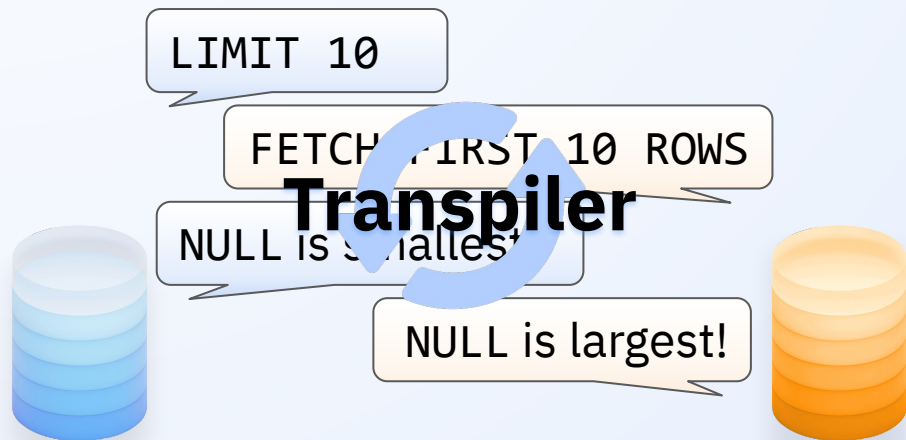
- **Complexity vs. validity tradeoff**

- Complex queries are essential for exposing deep bugs, but ensuring validity is difficult.

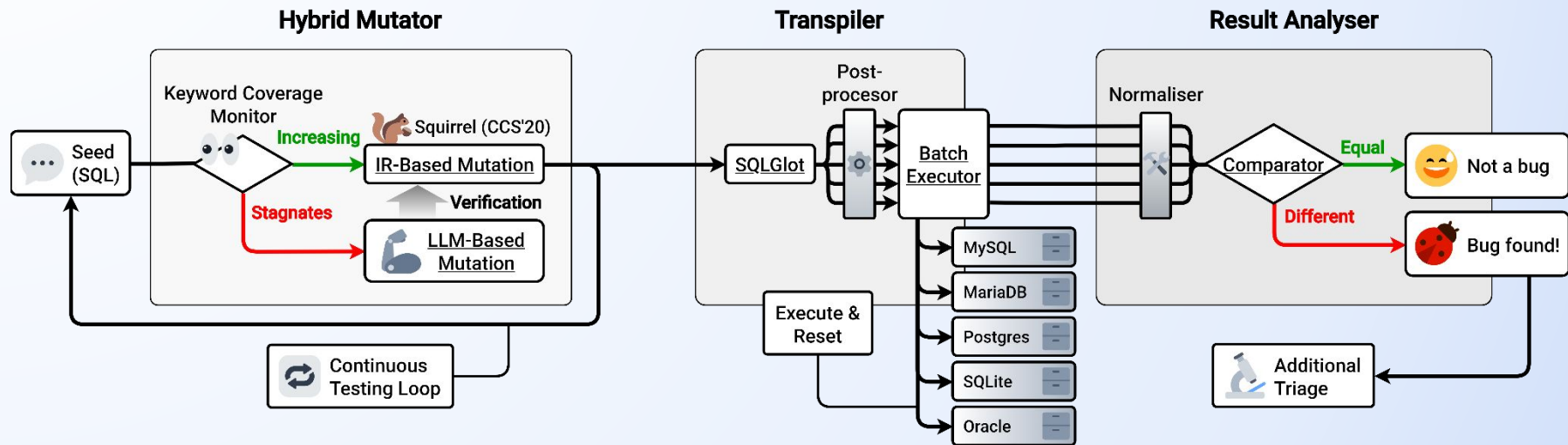


QueryHouse

- SQL discrepancy → **Transpiler**
- Complexity vs. validity tradeoff → **LLM-based mutation**



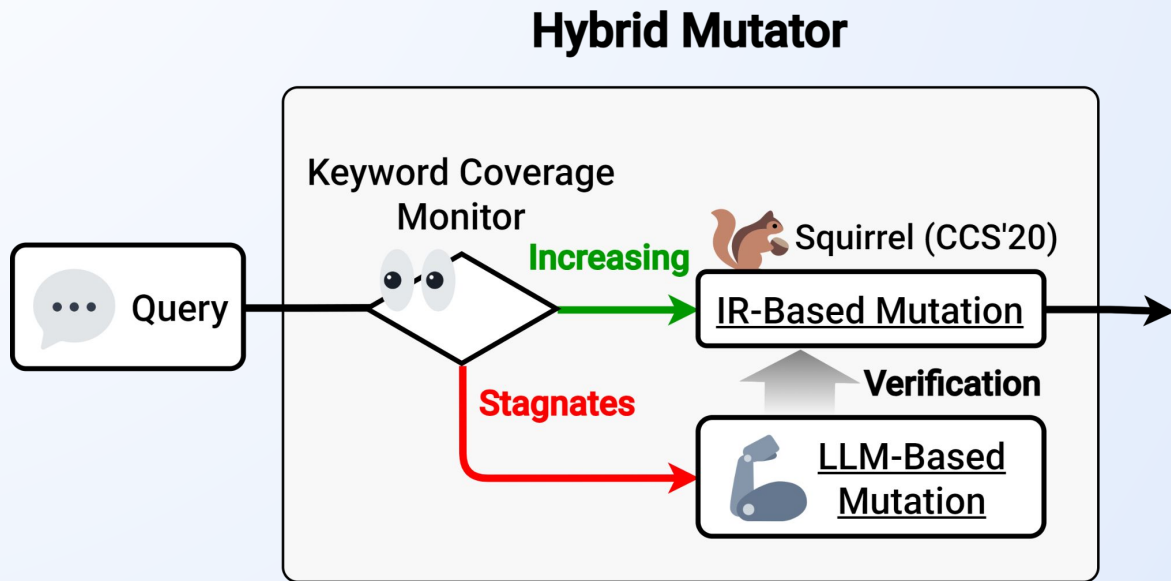
QueryHouse



Executive workflow of QueryHouse

QueryHouse

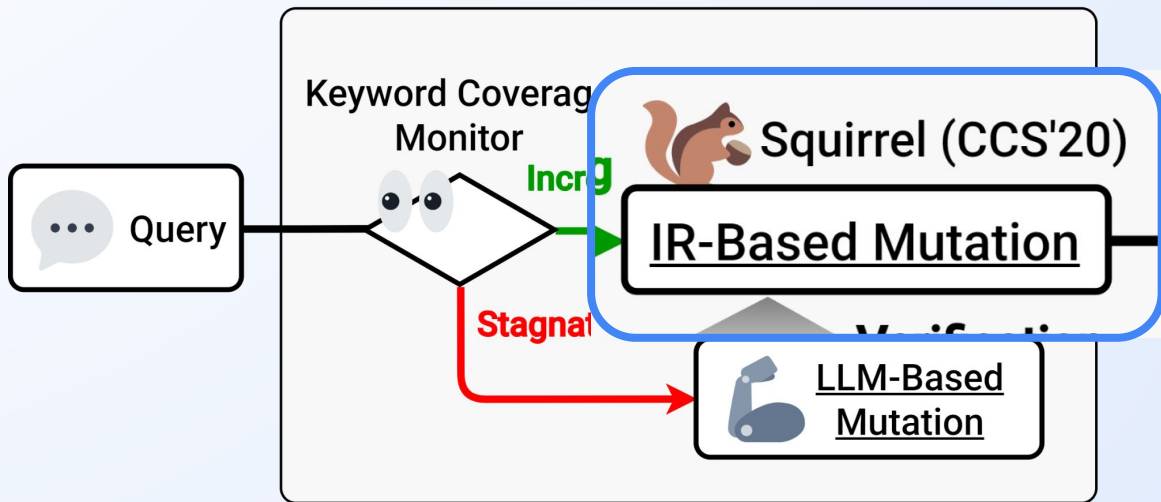
Hybrid Mutator



QueryHouse

Hybrid Mutator

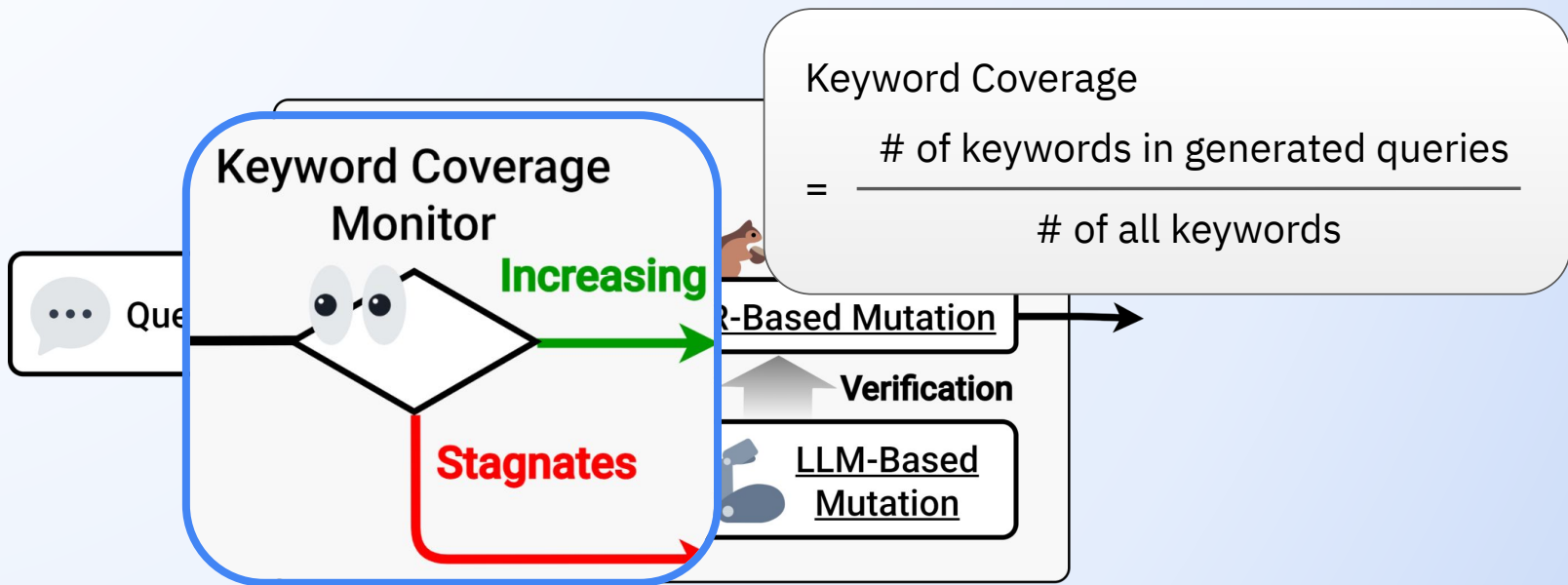
- Uses **IR-based mutation** for reliable exploration.



QueryHouse

Hybrid Mutator

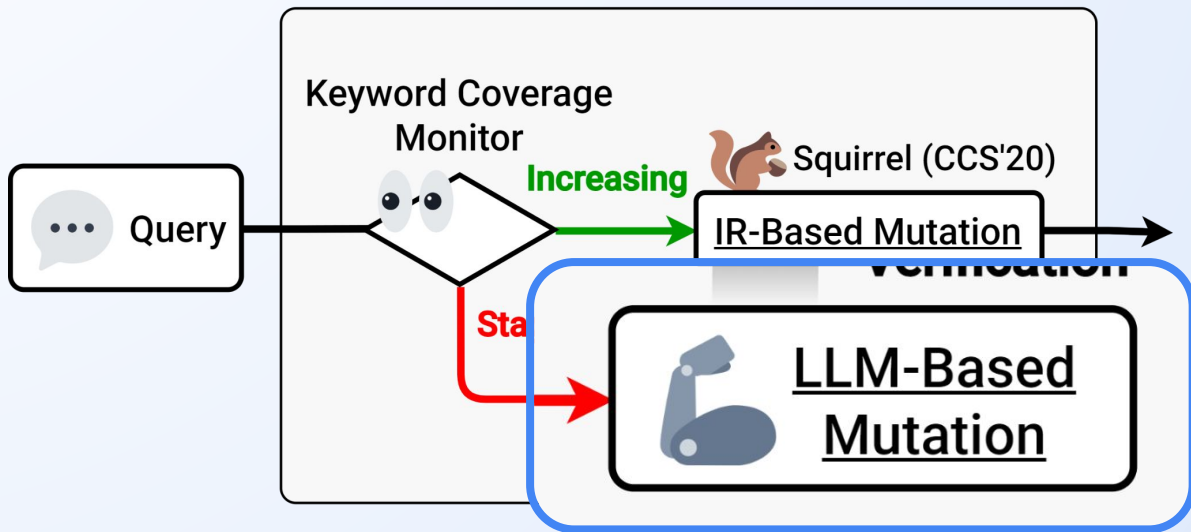
- Monitors stagnation with **keyword coverage**.



QueryHouse

Hybrid Mutator

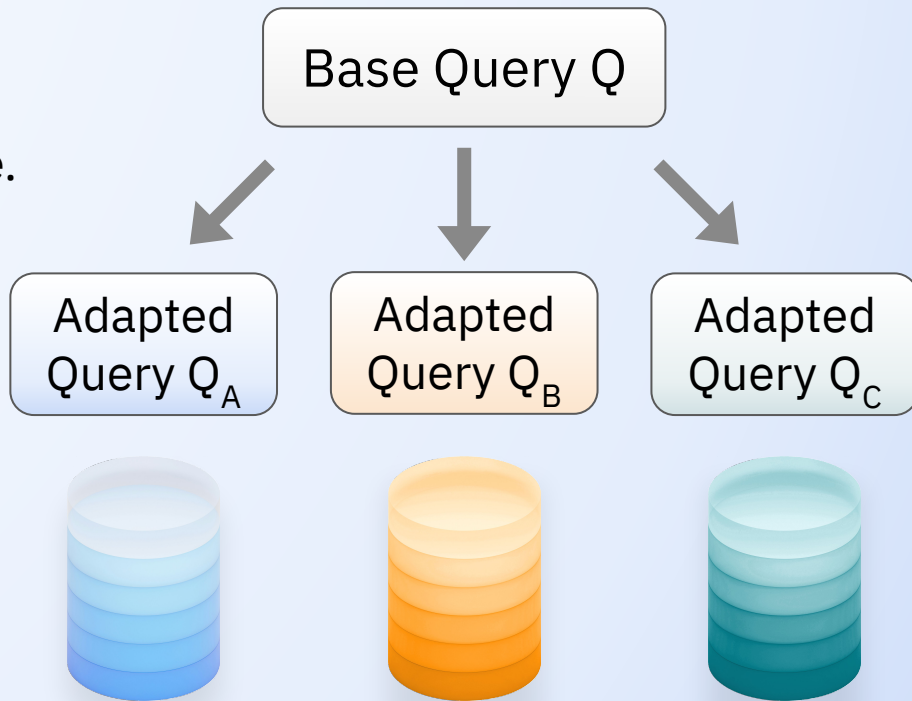
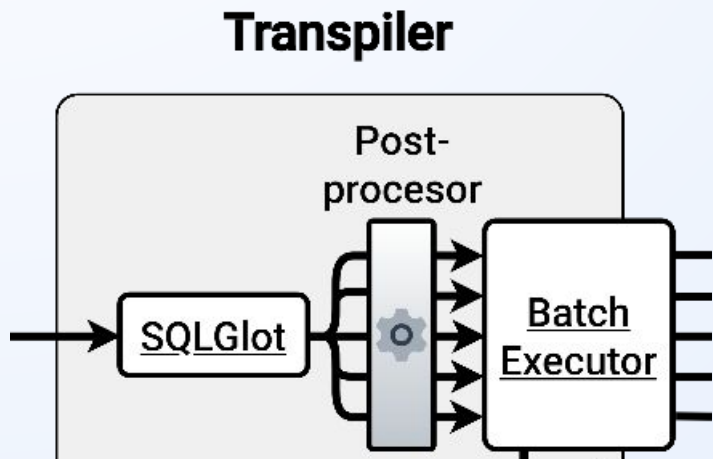
- Switches to **LLM-based mutator** when **exploration stalls**.



QueryHouse

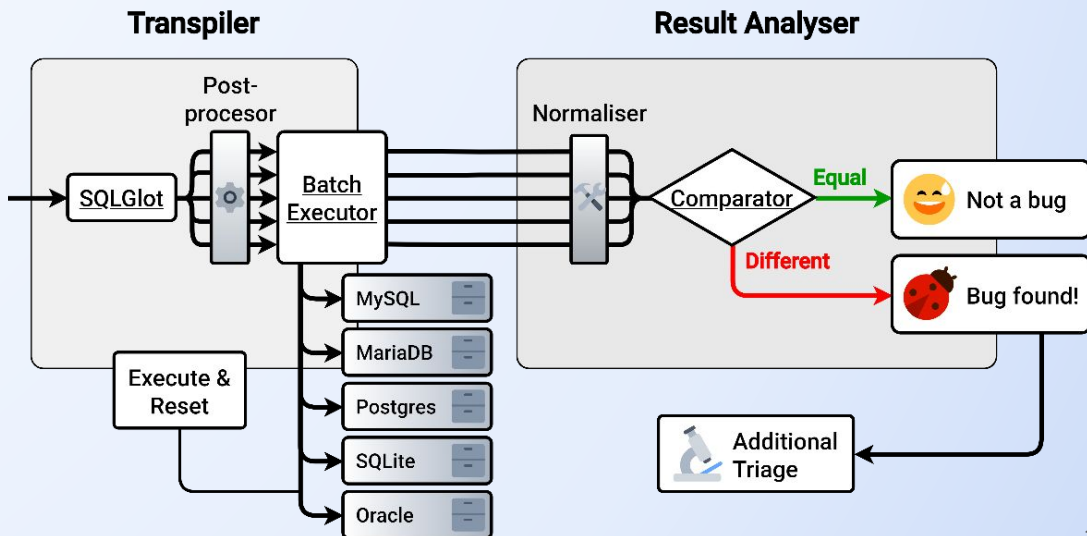
Transpiler

- Translates *a base query* into *adapted queries*.
- Preserves semantic equivalence.



Batch Executor & Result Analyzer

- **Executes** each adapted query in corresponding DBMSs in a batch.
- **Normalizes** the results and **compares** them.
- Discrepancies are flagged for further triage.



Bugs Found

Over a three-week campaign on five real-world DBMSs, QueryHouse found



11 Logic Bugs

- All verified
- 6 fixed

ID	DBMS	Description	Status
1	SQLite	FILTER not working in window function	Fixed
2	SQLite	REAL constant doubles	Fixed
3	SQLite	ROUND function overflow	Fixed
4	SQLite	SUBSTR function overflow	Fixed
5	MySQL	BETWEEN operator precedence mismatch	Verified
6	MySQL	IN operator precedence mismatch	Verified
7	MySQL	MAX as window aggregate returns NULL	Fixed
8	MariaDB	BETWEEN operator precedence mismatch	Fixed [†]
9	Oracle	LAG/LEAD function overflow	Verified
10	Oracle	NATURAL JOIN on same table results error	Verified
11	Oracle	ROUND function overflow	Verified

Logic bugs newly discovered by QueryHouse

Case study 1

SQLite REAL Constant Doubles

- Doubles specific range of REAL constant

```
SELECT 18446744073709551488;
```



Case study 2

Oracle LAG/LEAD Overflow

- Overflow in offset of LAG/LEAD window functions

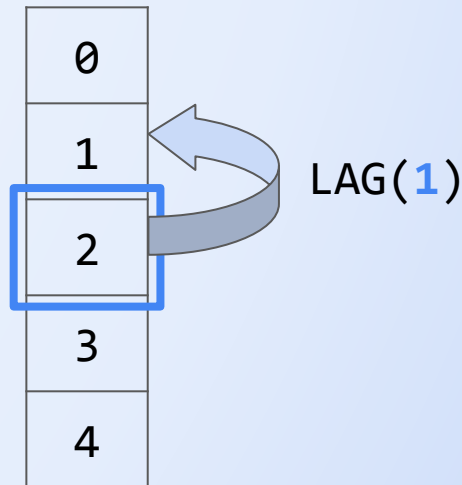
```
1 CREATE TABLE t0 (c0 INT);
2 INSERT INTO t0 VALUES (1), (2);
3 SELECT v1, LAG(v1, 18446744073709551614, -1)
   OVER (ORDER BY v1) FROM v0;
```

Expected 

SQL> V1	LAG(V1,...)OVER...
0	-1
1	-1
2	-1
3	-1
4	-1

Result 

SQL> V1	LAG(V1,...)OVER...
0	1
1	2
2	3
3	4
4	-1



Case study 2

Oracle LAG/LEAD Overflow

- Overflow in offset of LAG/LEAD window functions

```
1 CREATE TABLE t0 (c0 INT);
2 INSERT INTO t0 VALUES (1), (2);
3 SELECT v1, LAG(v1, 18446744073709551614, -1)
   OVER (ORDER BY v1) FROM v0;
```

Expected ✓

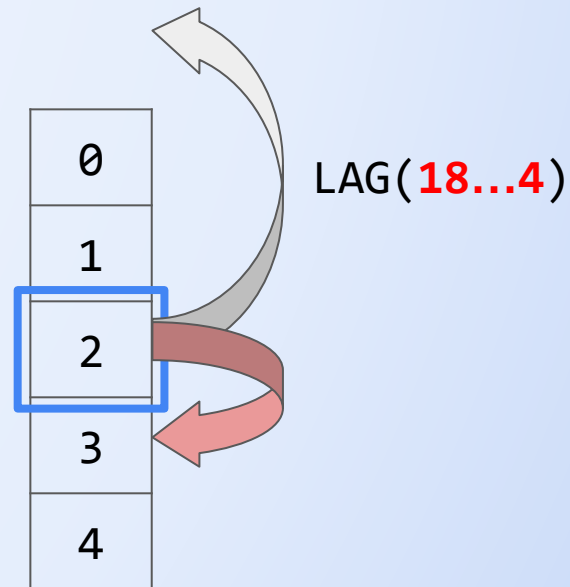
```
SQL> V1 LAG(V1,...)OVER...
```

0	-1
1	-1
2	-1
3	-1
4	-1

Result 🐞

```
SQL> V1 LAG(V1,...)OVER...
```

0	1
1	2
2	3
3	4
4	-1



Evaluation

Why Existing Approaches Miss Them

- **Sympathetic failure**

- A DBMS may **consistently return incorrect results** for transformed queries.
- E.g., fundamentally broken features

```
SELECT 18446744073709551488;
```



```
3.68934881474191e+19
```

- **Difficulty of designing an oracle**

- An oracle design for **complex and advanced features** is challenging.
- E.g., windows function.

```
CREATE TABLE t0 (c0 INT);  
INSERT INTO t0 VALUES (1), (2);  
SELECT v1, LAG(v1,  
18446744073709551614, -1)  
OVER (ORDER BY v1) FROM v0;
```

Why Existing Approaches Miss Them

- **Sympathetic failure**
 - A DBMS may **consistently return incorrect results** for transformed
- **Difficulty of designing an oracle**
 - An oracle design for **complex and advanced features** is challenging.

QueryHouse targets a **different failure** from existing approaches.

Evaluation

Transpilation and LLM Mutation

- **Transpiler improves validity** across multiple DBMSs.
 - $\text{valid}@k$: how many of test cases are valid across at least k DBMSs

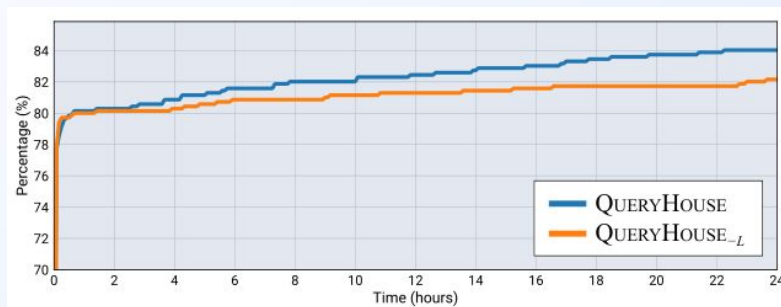
		DBMS					valid@k	
		SQLite	Oracle	MySQL	PSQL	Maria	valid@5	valid@2
with transpiler →	QUERYHOUSE	0.4883	0.2168	0.2495	0.2030	0.2747	0.1633	0.2795
without transpiler →	QUERYHOUSE _{-T}	0.4883	0.1740	0.1096	0.1660	0.0806	0.0457	0.2395
	Δ Validity	–	24.56%	127.64%	22.29%	241.05%	257.03%	16.72%

Validity with/without transpiler

Evaluation

Transpilation and LLM Mutation

- **Transpiler improves validity** across multiple DBMSs.
- **LLM-based mutation** helps **escape exploration plateaus**.



← with LLM-based mutation
← without LLM-based mutation

Keyword coverage with/without LLM

Evaluation

Transpilation and LLM Mutation

- **Transpiler improves validity** across multiple DBMSs.

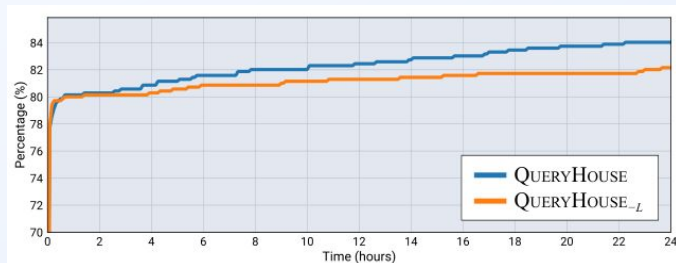
with transpiler →

without transpiler →

	DBMS					valid@k	
	SQLite	Oracle	MySQL	PSQL	Maria	valid@5	valid@2
QUERYHOUSE	0.4883	0.2168	0.2495	0.2030	0.2747	0.1633	0.2795
QUERYHOUSE _{-T}	0.4883	0.1740	0.1096	0.1660	0.0806	0.0457	0.2395
Δ Validity	–	24.56%	127.64%	22.29%	241.05%	257.03%	16.72%

Validity with/without transpiler

- **LLM-based mutation helps escape exploration plateaus.**



Keyword coverage with/without LLM

← with LLM-based mutation

← without LLM-based mutation

Evaluation

Transpilation and LLM Mutation

- **Transpiler improves validity** across multiple DBMSs.

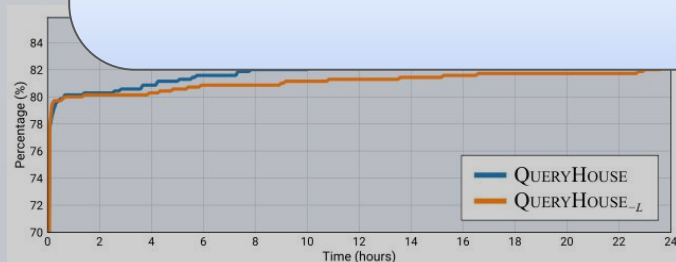
	DBMS					valid@k	
	SQLite	Oracle	MySQL	PSQL	Maria	valid@5	valid@2
						0.2795	
						0.2395	
						16.72%	

Transpiler and LLM-based mutation makes

differential testing practical

by improving validity and exploration.

- LLM



← with LLM-based mutation

← without LLM-based mutation

Future Work



Better validity

Improve the **mutator** and **transpiler** to generate more valid and complex queries.



More targeted exploration

Go beyond keyword coverage with stronger **LLM mutation** and **triggering strategies**.

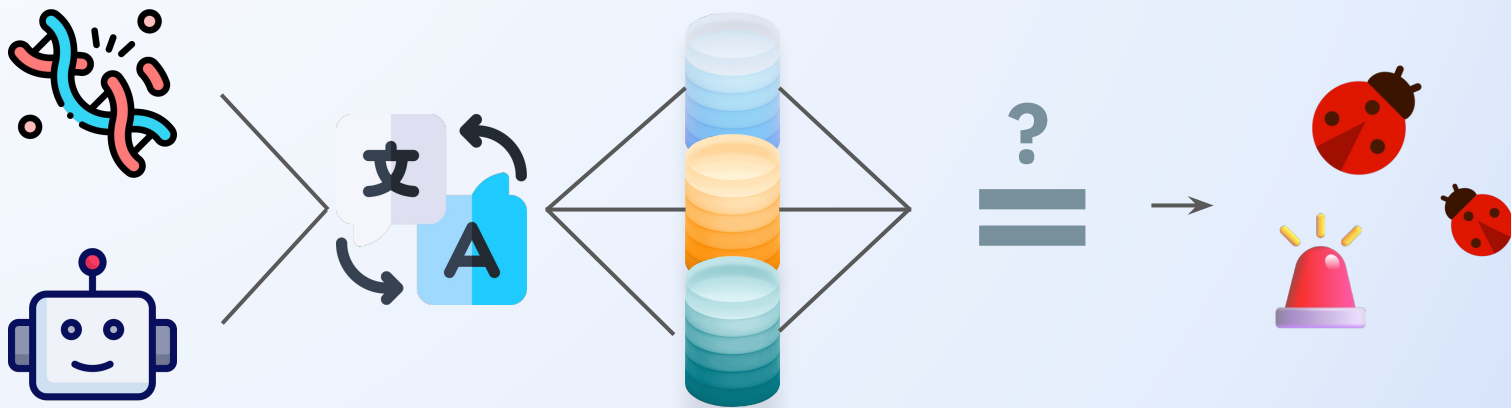


Scalable analysis

Automate triage and deduplication to reduce manual effort.

Conclusion

- **Differential testing** breaks a fundamental oracle blind spot.
- QueryHouse implements it with **transpilation** + **hybrid LLM mutation**.
- We evaluated it on 5 DBMSs, found **11 new logic bugs**.



Thank you 🐱

