Bespoke-Card: Why Tune When You Can Generate? Synthesizing Workload-Specific Cardinality Estimators



Johannes Wehrstein, Anton Winter, Timo Eckmann, Carsten Binnig

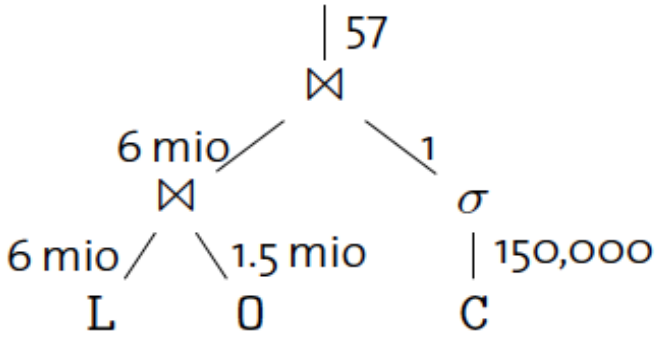
Technical University Darmstadt, Germany

SYSTEMS

Best Paper Award

Query Optimization in a nutshell

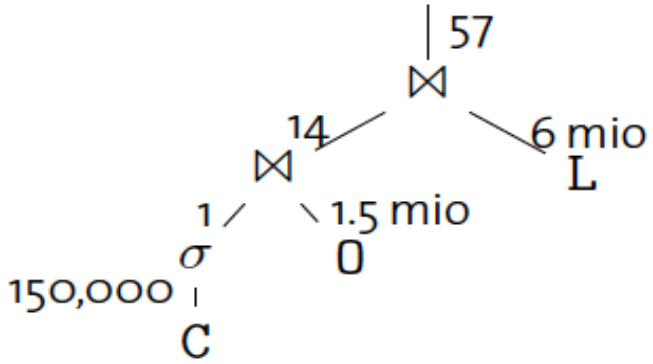
Determines join-orders, physical operator selections…



$(L \bowtie O) \bowtie C$

Intermediate Cardinalities:

13.65 M



$(C \bowtie O) \bowtie L$

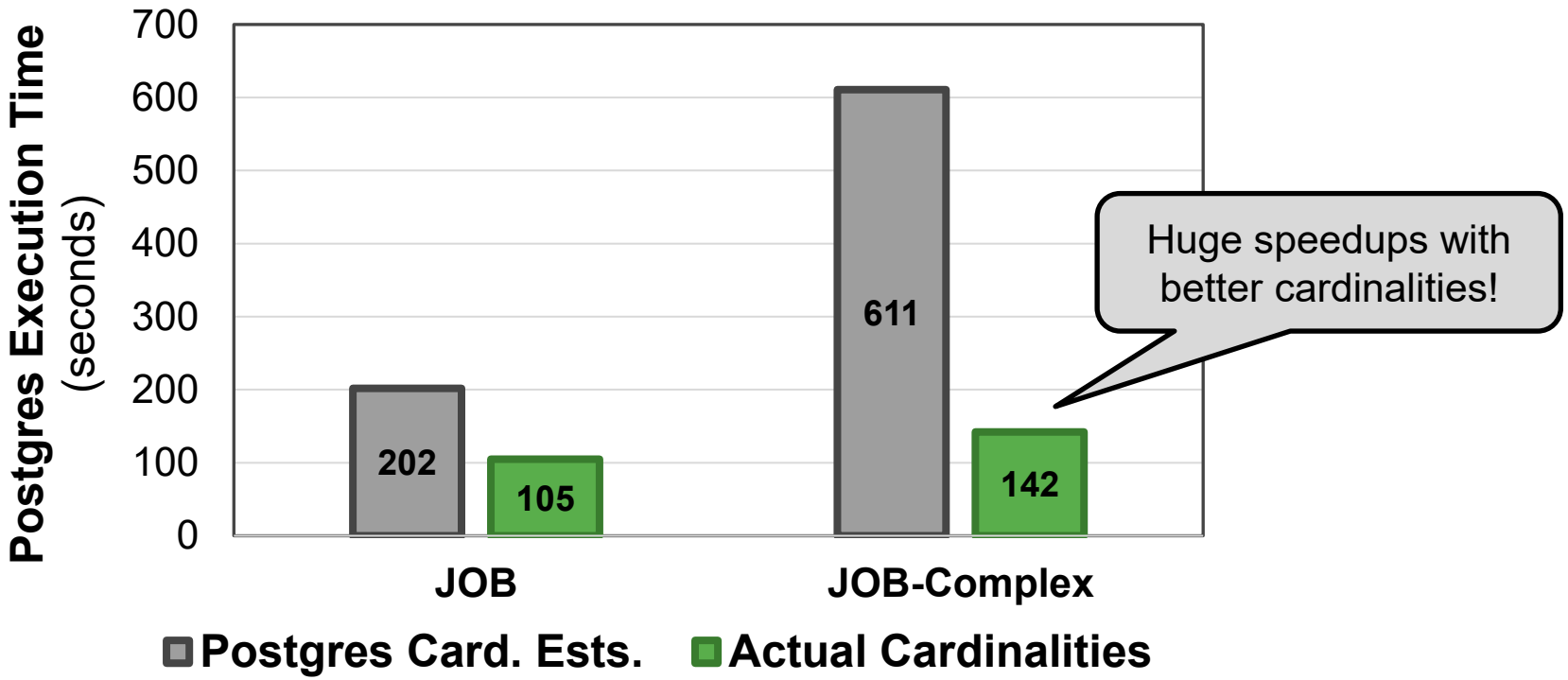
Intermediate Cardinalities:

7.65 M

Decision between plans decided by cardinality estimates

Cardinality Estimation sits at the core of Query-Optimization

Accurate Cardinalities Are Important



What is the problem with existing cardinality estimators?

Pitfalls of existing approaches

Independence assumption:

City	Country
Darmstadt	Germany
Darmstadt	USA
Paris	France
London	GB
…	…

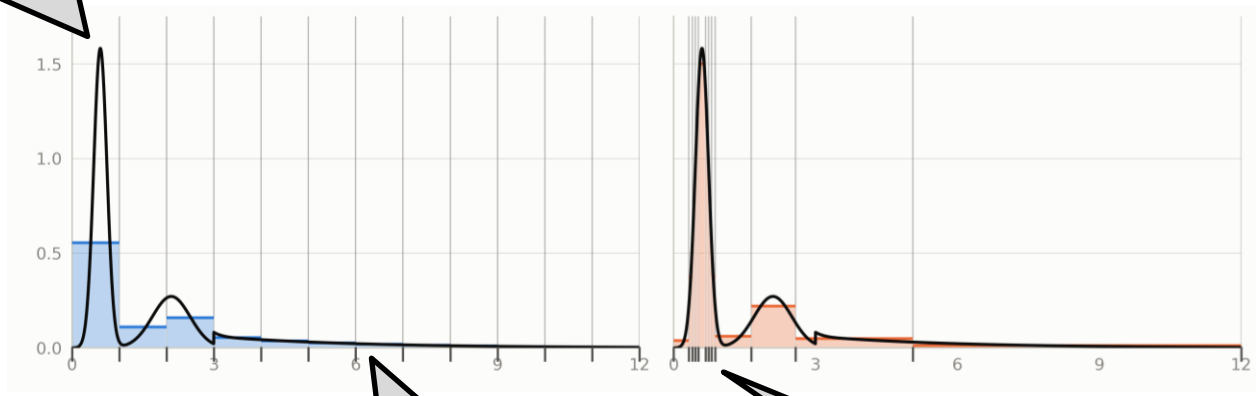
Columns are correlated!
Traditional 1-D Hists / … assume independence

Pre-sized statistics:

Not capturing peak

Equi-Width Histogram (12 bins)

Optimal Binning (12 bins)



Useless bins

Bins sized to where it matters

Classical approaches are inaccurate

Existing Estimation Methods

Classical Approaches (synopses and sampling)

- 👍 Fast
- 🆗 Limited accuracy (simplifying assumptions)
- 👎 General purpose

Can't adapt to the workload!

Learned Approaches (data-driven, wl-driven, foundational)

- 👍 Higher Accuracy (tuned to wl)
- 👎 Expensive & slow
- 👎 Very limited (e.g. mostly cannot handle strings)

In practice hard to use (limited SQL, too costly)

Classical approaches still dominating production

Bespoke-Card: The best of two worlds

Low-overhead

of classical estimators

+

Accuracy

of wl-tuned learned estimators

Workload-specific ("Bespoke") Classical Estimators

Histograms with optimized bin boundaries

Cross-Column statistics for cols queried together

Column/Predicate specific sampling methods

Workload-aware substring / filter stats

Synthesizing Bespoke-Cardinality Estimators

Workload Information

+

Database Statistics

Bespoke Cardinality Estimator

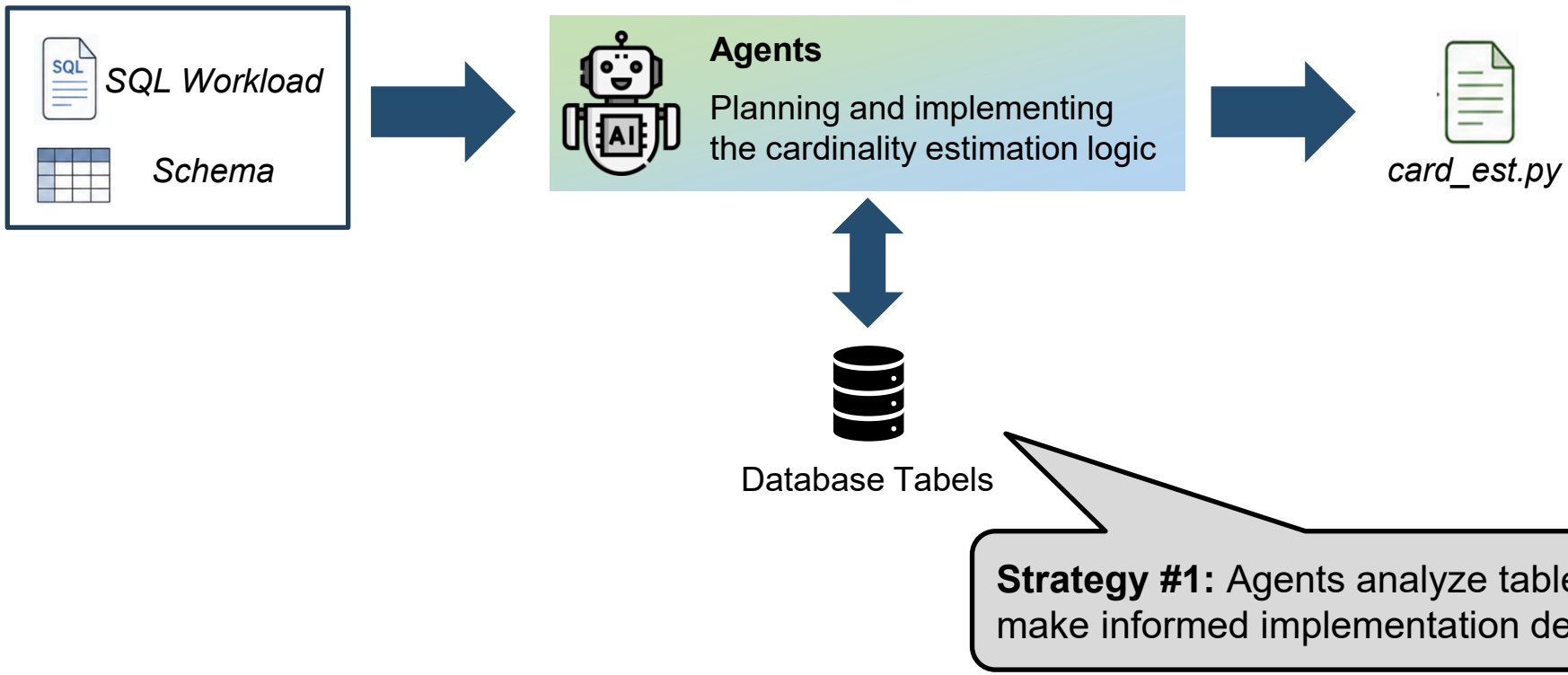
```
def populate_stats(self, tables: list):
    # populate bespoke histograms
    # perform bespoke sampling

def estimate(self, tables: list,
             filters: list, joins: list) -> int:
    # query hist / sampling / … logic
    est_rows =…

    return est_rows
```

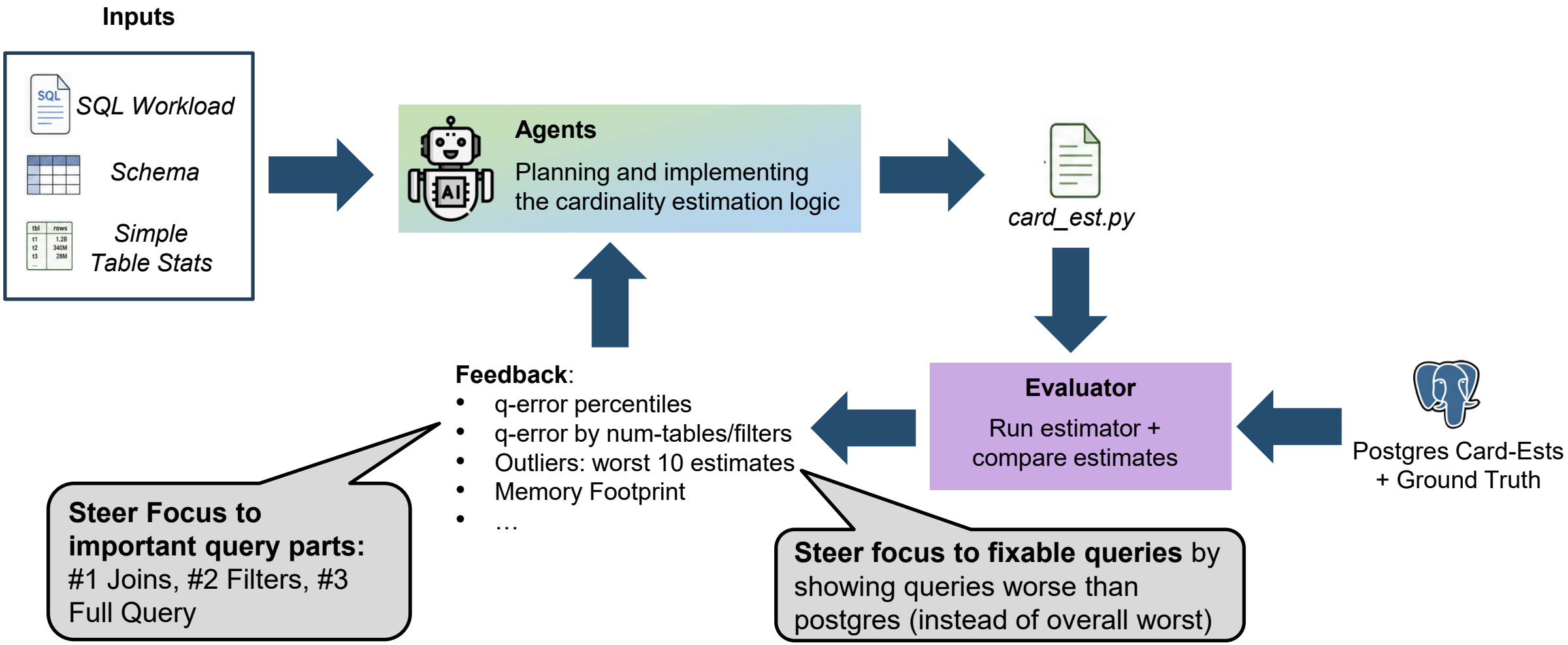
Statistics, hists, … designed for the workload!

Bespoke-Card: Approach



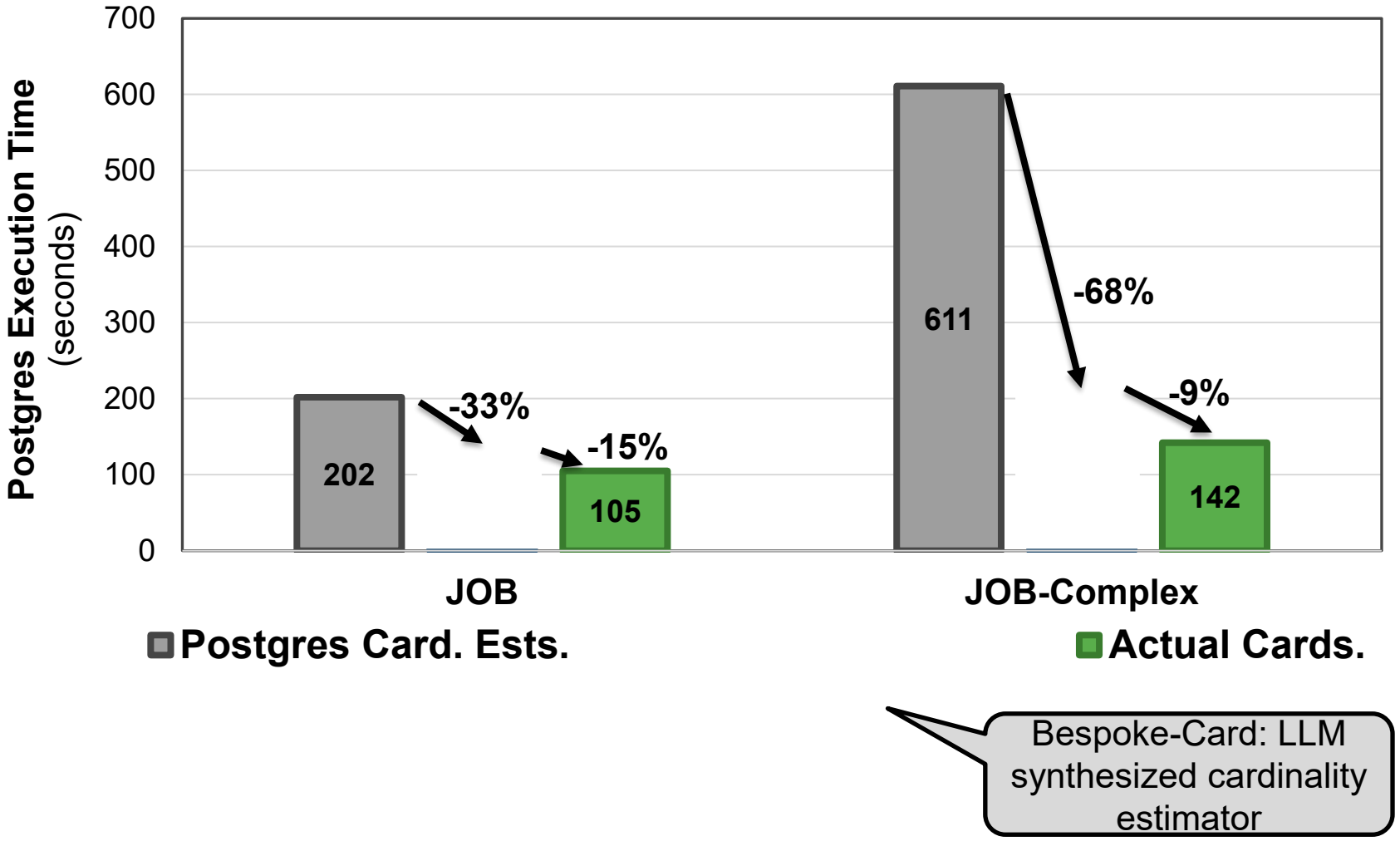
Agents implement estimator with distributions in mind

Strategy #2: Targeted Feedback (optional)



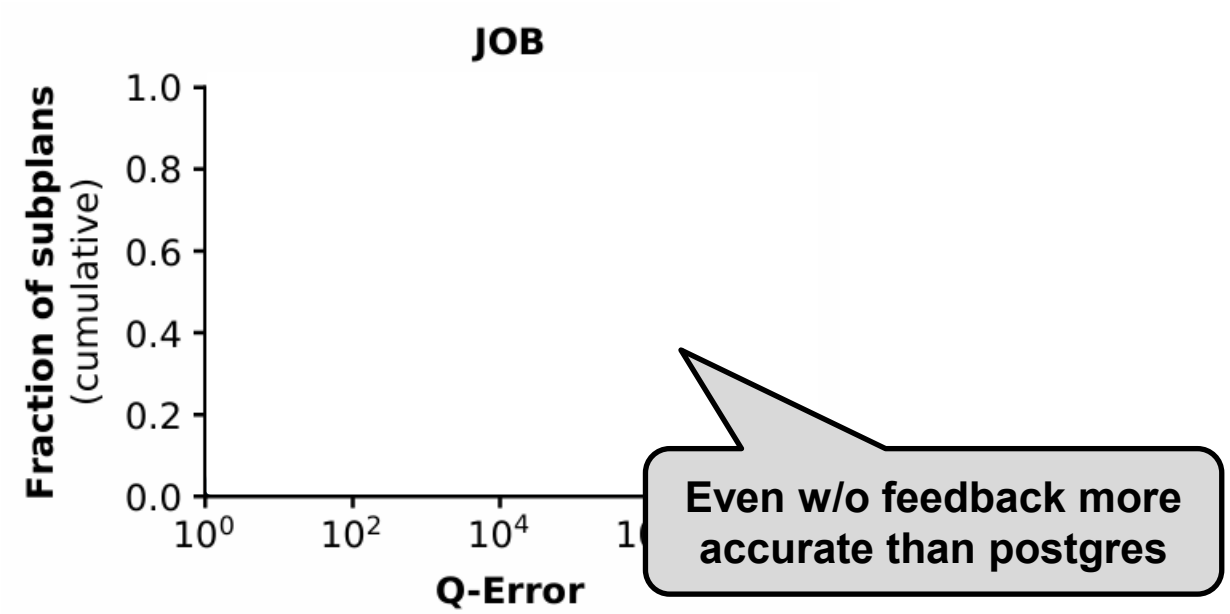
Targeted feedback ensures forward progress

Bespoke-Card



Bespoke-Card reduces runtime by 33% (JOB), 68% (JOB-Complex)

Estimation Accuracy: Q-Error



WL	System	q5	q50	q95
JOB	PostgreSQL (16)	1.50	190.5	132k
	Bespoke-Card	1.24	11.5	1k

Improvements increase in tail distribution

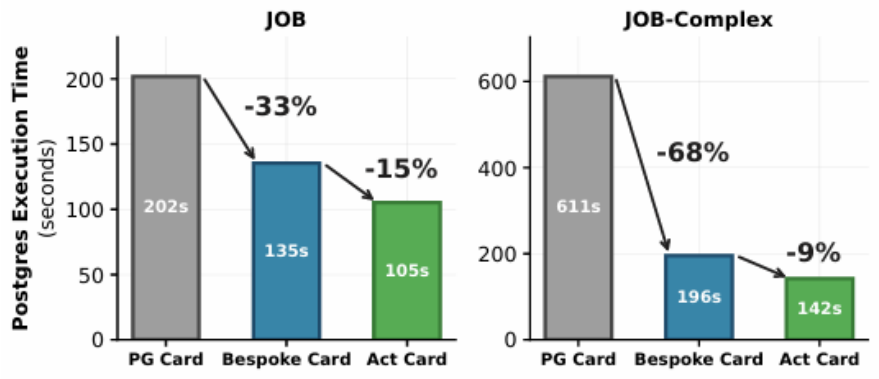
Synthesis Cost & Time

Initial Implementation (JOB):
- 25 LLM Turns
- $ 0.67
- <10mins

Feedback Stages (JOB):
- 63 LLM Turns
- $ 9.07
- <60mins

Q-Errors orders of magnitude lower for less than $10

Bespoke-Card



- **Workload**-aware **classical** cardinality estimators are orders of magnitude **more accurate** than PG
- **LLM Agents** can effectively design and implement bespoke-cardinality estimators (**<$10**)
- End-to-end runtimes reduce drastically (-33% on JOB, -68% on JOB-Complex)

Learn more:
github.com/DataManagementLab/BespokeCard
- Paper
- Source code
- Synthesized Engines

Synthesize full DB Engines?

Tuesday Research 2 (Cloud-Native Data Systems)
"Bespoke-OLAP"

SynnoDB synthesizes workload-specific database systems that deliver orders-of-magnitude performance improvements.

synnodb.com