AutoSteer: Learned Query Optimization for Any SQL Database

Christoph Anneser[1,2], Nesime Tatbul[2,3], David Cohen[2], Zhenggang Xu[4]
Prithviraj Pandian[4], Nikolay Laptev[4], and Ryan Marcus[5]

[1]TUM, [2]Intel, [3]MIT, [4]Meta, [5]University of Pennsylvania

VLDB, August 30, 2023

TUM Uhrenturm

Background – Steering Query Optimizers

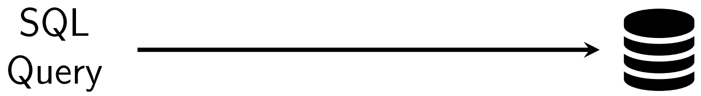
Many Database Management Systems expose tunable optimizer knobs.

- Usually belong to *rewrite rules* of the rule-based optimizer
- Can be used to *steer* query optimization

Background – Steering Query Optimizers

Many Database Management Systems expose tunable optimizer knobs.

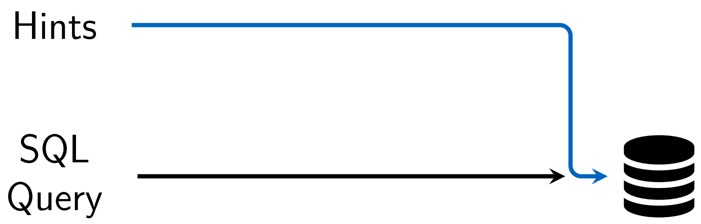
- Usually belong to *rewrite rules* of the rule-based optimizer
- Can be used to *steer* query optimization



Background – Steering Query Optimizers

Many Database Management Systems expose tunable optimizer knobs.

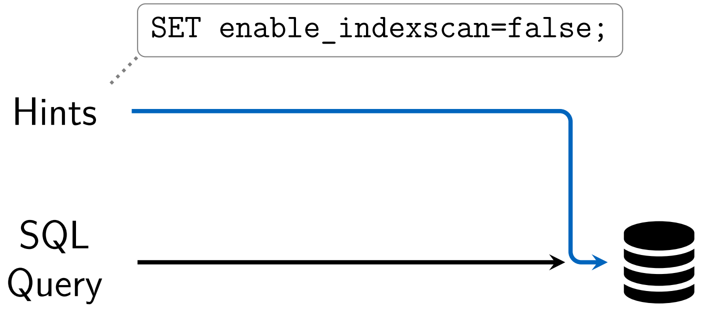
- Usually belong to *rewrite rules* of the rule-based optimizer
- Can be used to *steer* query optimization



Background – Steering Query Optimizers

Many Database Management Systems expose tunable optimizer knobs.

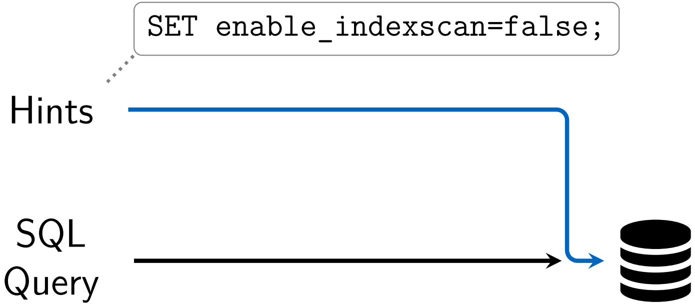
- Usually belong to *rewrite rules* of the rule-based optimizer
- Can be used to *steer* query optimization



Background – Steering Query Optimizers

Many Database Management Systems expose tunable optimizer knobs.

- Usually belong to *rewrite rules* of the rule-based optimizer
- Can be used to *steer* query optimization

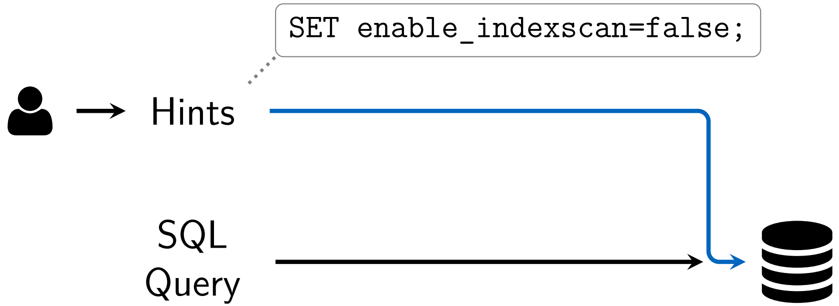


Hint-sets combine multiple hints. For example: `{indexscan:false, nestloop:false}`

Background – Steering Query Optimizers

Many Database Management Systems expose tunable optimizer knobs.

- Usually belong to *rewrite rules* of the rule-based optimizer
- Can be used to *steer* query optimization

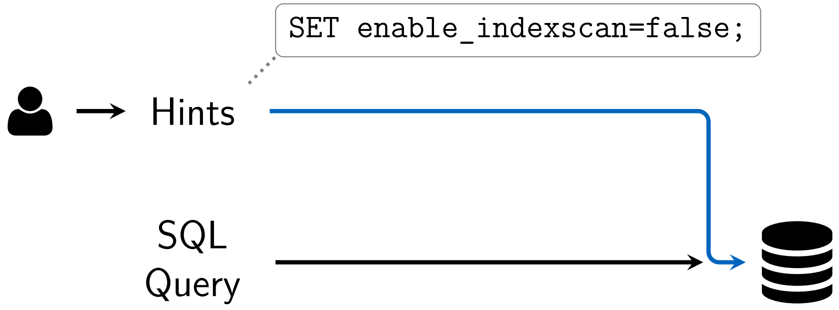


Hint-sets combine multiple hints. For example: `{indexscan:false, nestloop:false}`

Background – Steering Query Optimizers **Manually**

Many Database Management Systems expose tunable optimizer knobs.

- Usually belong to *rewrite rules* of the rule-based optimizer
- Can be used to *steer* query optimization

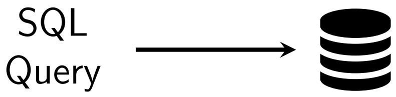


Hint-sets combine multiple hints. For example: `{indexscan:false, nestloop:false}`

Background – Steering Query Optimizers **Automatically**

Lot of recent work on steered query optimizers:

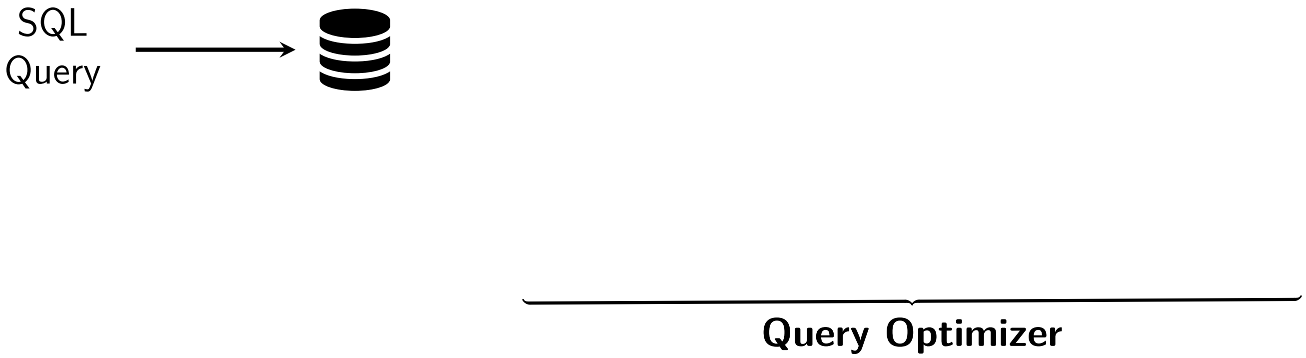
- SIGMOD'21: "Bao: Learning to steer query optimizers", Marcus et al.
- SIGMOD'21: "Steering Query Optimizers: A Practical Take on Big Data Workloads", Negi et al. [Industry]
- SIGMOD'22: "Deploying a Steered Query Optimizer in Production at Microsoft", Zhang et al. [Industry]



Background – Steering Query Optimizers **Automatically**

Lot of recent work on steered query optimizers:

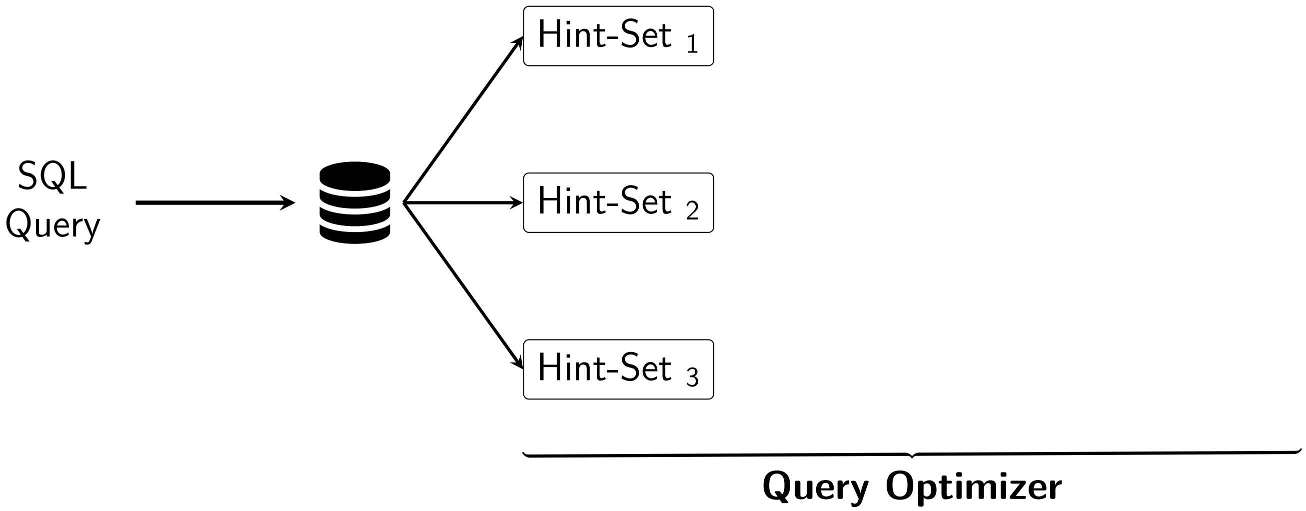
- SIGMOD'21: "Bao: Learning to steer query optimizers", Marcus et al.
- SIGMOD'21: "Steering Query Optimizers: A Practical Take on Big Data Workloads", Negi et al. [Industry]
- SIGMOD'22: "Deploying a Steered Query Optimizer in Production at Microsoft", Zhang et al. [Industry]



Background – Steering Query Optimizers **Automatically**

Lot of recent work on steered query optimizers:

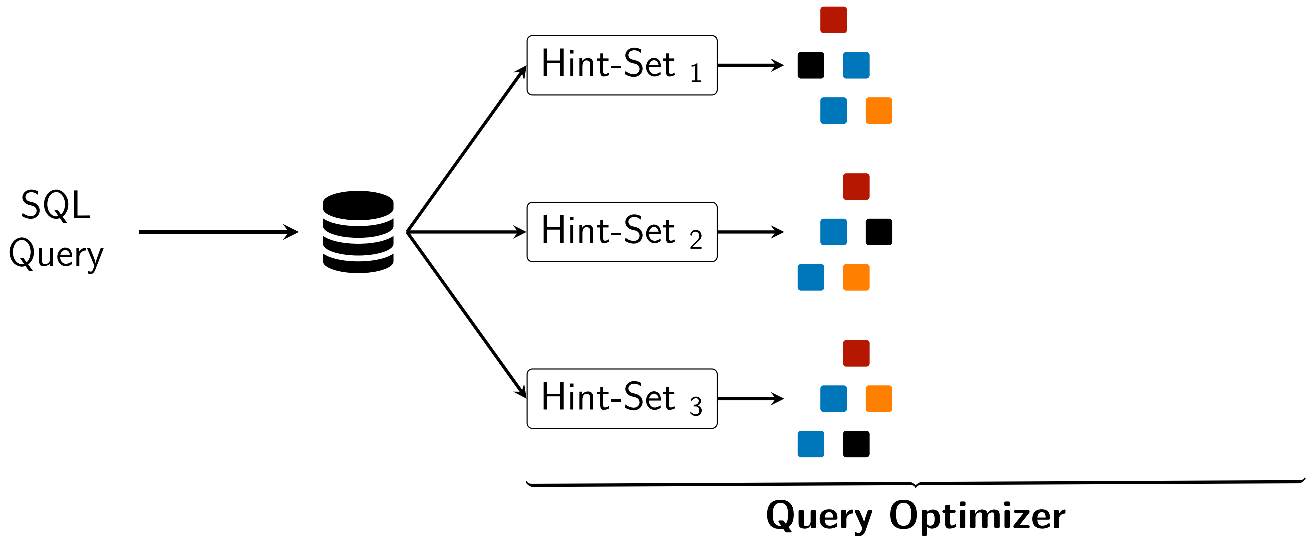
- SIGMOD'21: "Bao: Learning to steer query optimizers", Marcus et al.
- SIGMOD'21: "Steering Query Optimizers: A Practical Take on Big Data Workloads", Negi et al. [Industry]
- SIGMOD'22: "Deploying a Steered Query Optimizer in Production at Microsoft", Zhang et al. [Industry]



Background – Steering Query Optimizers **Automatically**

Lot of recent work on steered query optimizers:

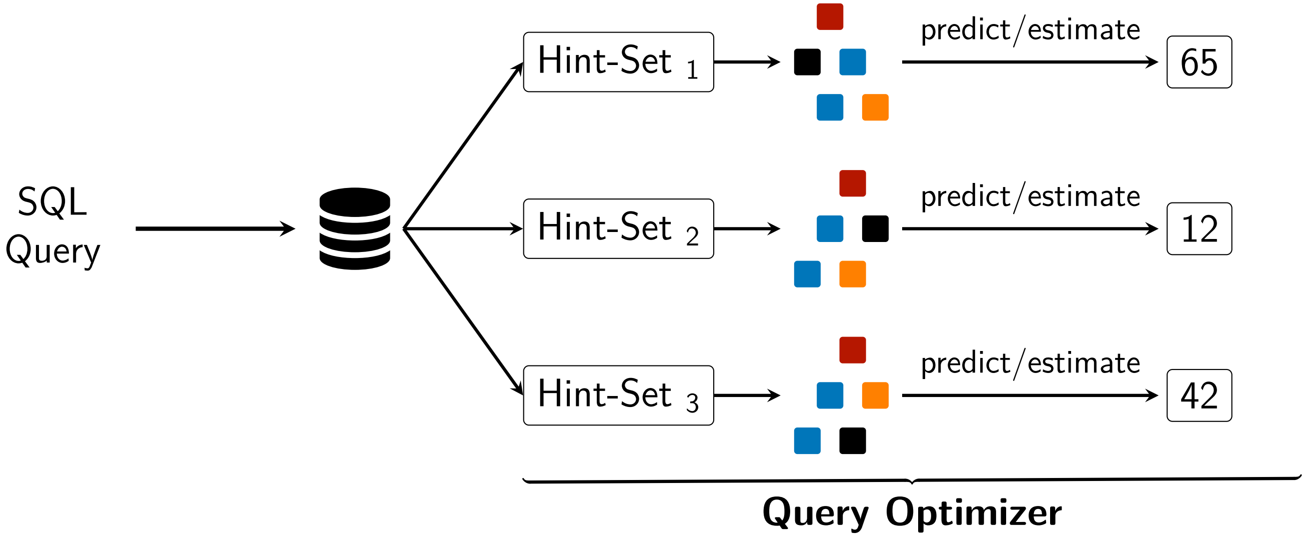
- SIGMOD'21: "Bao: Learning to steer query optimizers", Marcus et al.
- SIGMOD'21: "Steering Query Optimizers: A Practical Take on Big Data Workloads", Negi et al. [Industry]
- SIGMOD'22: "Deploying a Steered Query Optimizer in Production at Microsoft", Zhang et al. [Industry]



Background – Steering Query Optimizers **Automatically**

Lot of recent work on steered query optimizers:

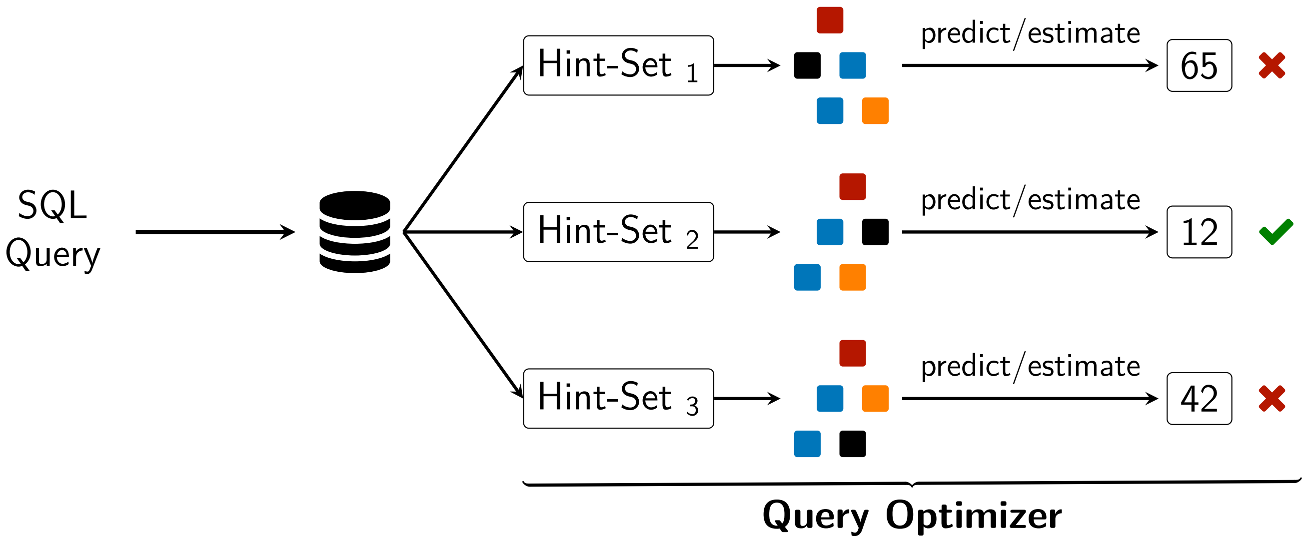
- SIGMOD'21: "Bao: Learning to steer query optimizers", Marcus et al.
- SIGMOD'21: "Steering Query Optimizers: A Practical Take on Big Data Workloads", Negi et al. [Industry]
- SIGMOD'22: "Deploying a Steered Query Optimizer in Production at Microsoft", Zhang et al. [Industry]



Background – Steering Query Optimizers **Automatically**

Lot of recent work on steered query optimizers:

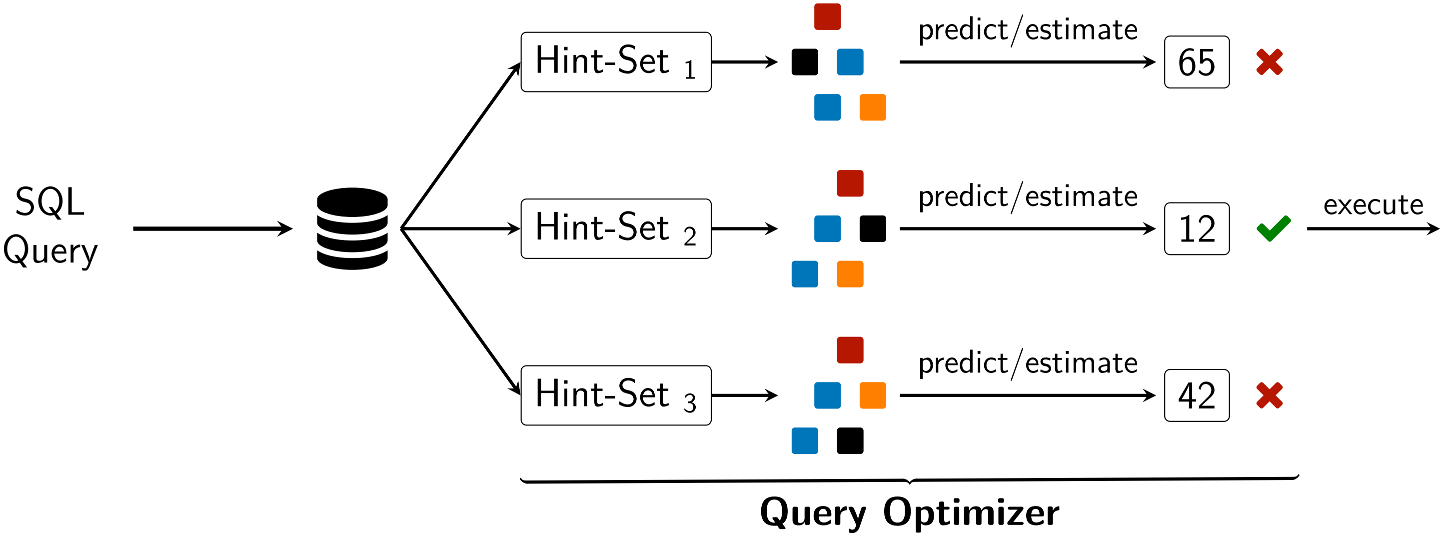
- SIGMOD'21: "Bao: Learning to steer query optimizers", Marcus et al.
- SIGMOD'21: "Steering Query Optimizers: A Practical Take on Big Data Workloads", Negi et al. [Industry]
- SIGMOD'22: "Deploying a Steered Query Optimizer in Production at Microsoft", Zhang et al. [Industry]



Background – Steering Query Optimizers **Automatically**

Lot of recent work on steered query optimizers:

- SIGMOD'21: "Bao: Learning to steer query optimizers", Marcus et al.
- SIGMOD'21: "Steering Query Optimizers: A Practical Take on Big Data Workloads", Negi et al. [Industry]
- SIGMOD'22: "Deploying a Steered Query Optimizer in Production at Microsoft", Zhang et al. [Industry]



Limitations of Previous Approaches

✖ Databases usually expose up to hundreds of knobs

$\Rightarrow$ static, predefined *or* random hint-sets

Limitations of Previous Approaches

- ✖ Databases usually expose up to hundreds of knobs
  $\Rightarrow$ static, predefined *or* random hint-sets

- ✖ Tight integration into the DBMS' query optimizer

Limitations of Previous Approaches

- ✖ Databases usually expose up to hundreds of knobs
  $\Rightarrow$ static, predefined *or* random hint-sets
- ✖ Tight integration into the DBMS' query optimizer
- ✖ Requires good knowledge of the query optimizer

Limitations of Previous Approaches

- ✖ Databases usually expose up to hundreds of knobs
  $\Rightarrow$ static, predefined *or* random hint-sets
- ✖ Tight integration into the DBMS' query optimizer
- ✖ Requires good knowledge of the query optimizer
- ✖ Custom for every database

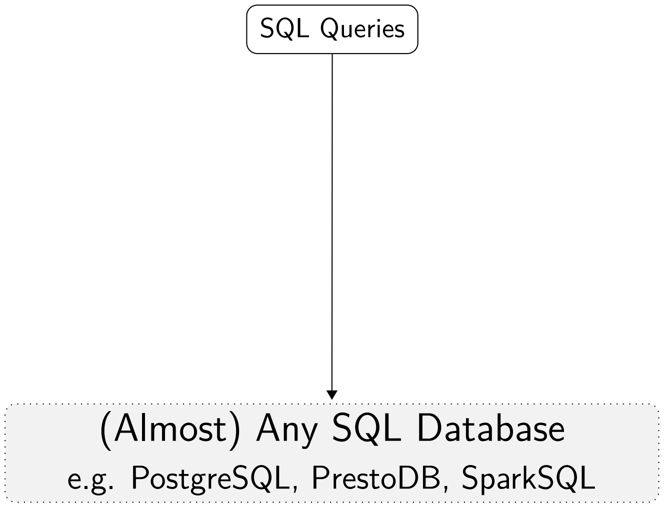
Limitations of Previous Approaches

- ✖ Databases usually expose up to hundreds of knobs
  $\Rightarrow$ static, predefined *or* random hint-sets
- ✖ Tight integration into the DBMS' query optimizer
- ✖ Requires good knowledge of the query optimizer
- ✖ Custom for every database

$\Rightarrow$ **AutoSteer is a generic framework to steer query optimizers!**

Limitations of Previous Approaches

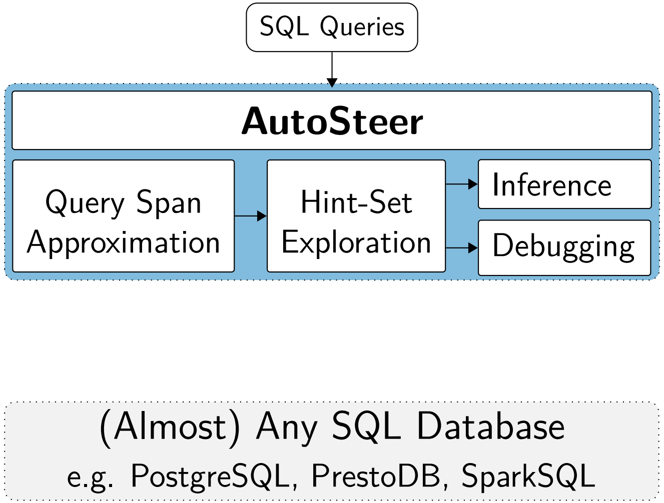
- ✖ Databases usually expose up to hundreds of knobs
  $\Rightarrow$ static, predefined *or* random hint-sets
- ✖ Tight integration into the DBMS' query optimizer
- ✖ Requires good knowledge of the query optimizer
- ✖ Custom for every database



$\Rightarrow$ **AutoSteer is a generic framework to steer query optimizers!**

Limitations of Previous Approaches

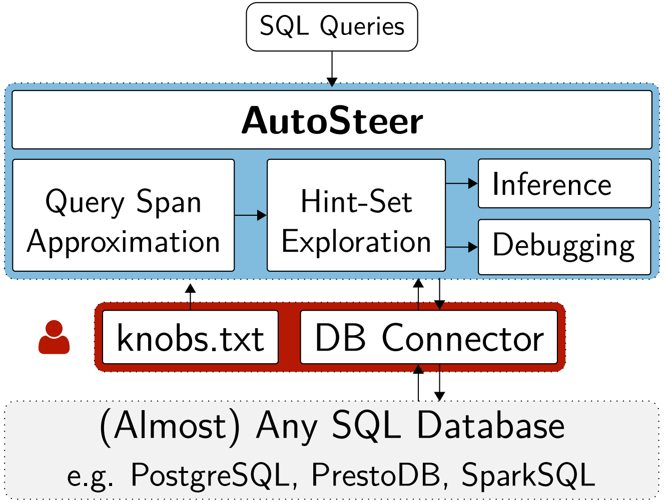
- ✖ Databases usually expose up to hundreds of knobs
  ⇒ static, predefined *or* random hint-sets
- ✖ Tight integration into the DBMS' query optimizer
- ✖ Requires good knowledge of the query optimizer
- ✖ Custom for every database



⇒ AutoSteer is a generic framework to steer query optimizers!

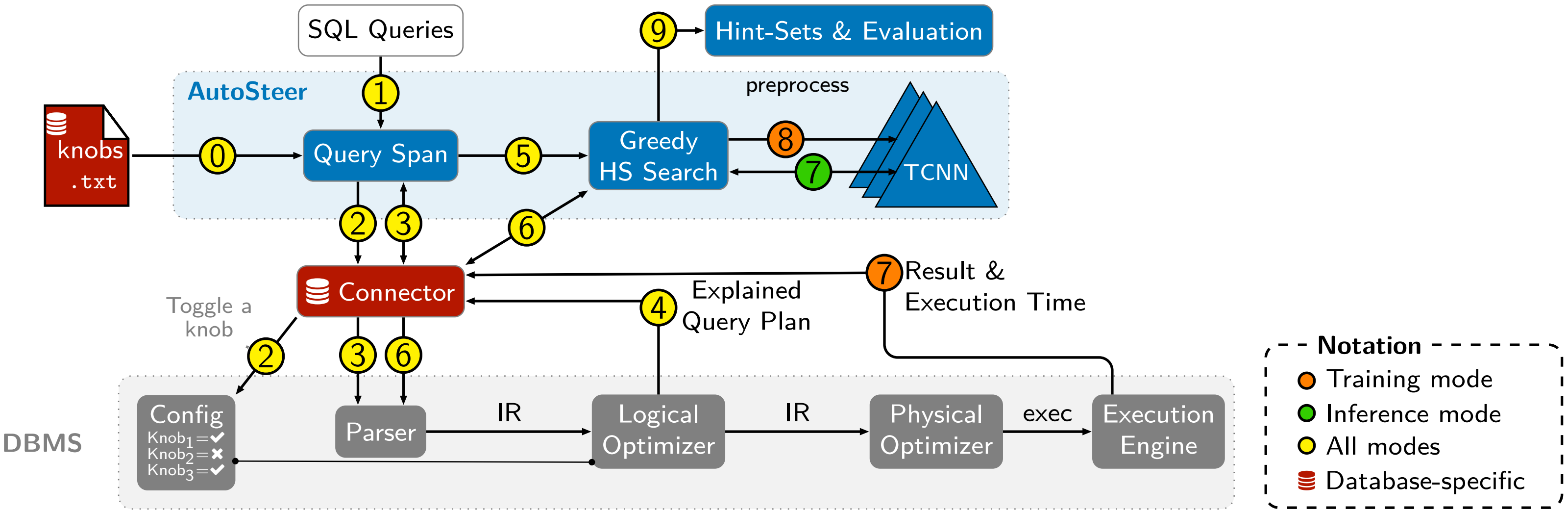
Limitations of Previous Approaches

- ✖ Databases usually expose up to hundreds of knobs
  $\Rightarrow$ static, predefined *or* random hint-sets
- ✖ Tight integration into the DBMS' query optimizer
- ✖ Requires good knowledge of the query optimizer
- ✖ Custom for every database

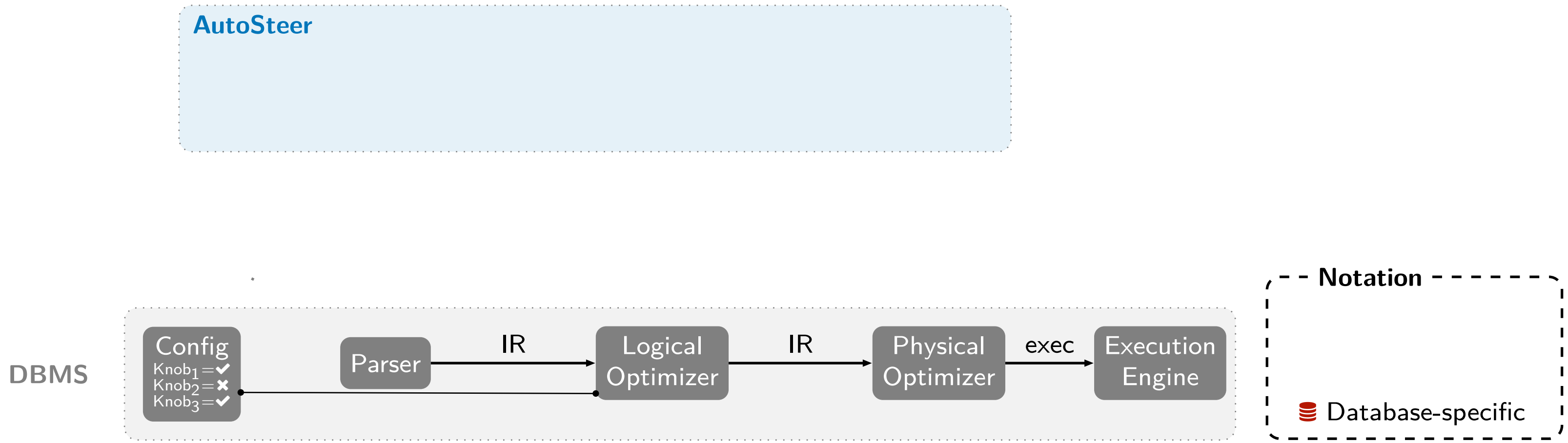


$\Rightarrow$ **AutoSteer is a generic framework to steer query optimizers!**

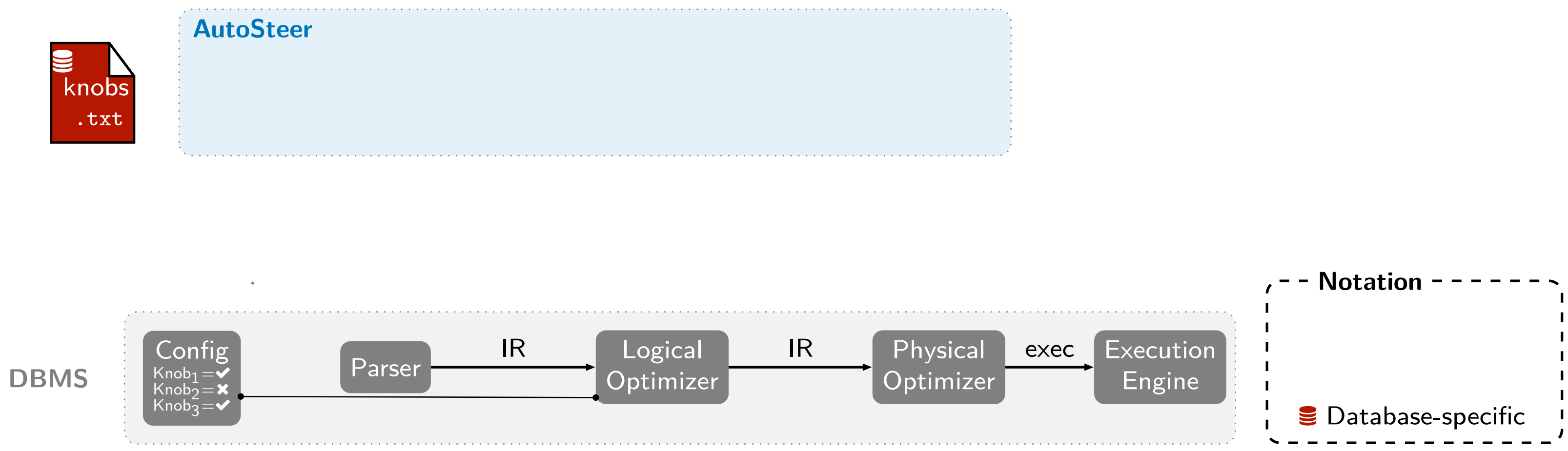
AutoSteer at a Glance



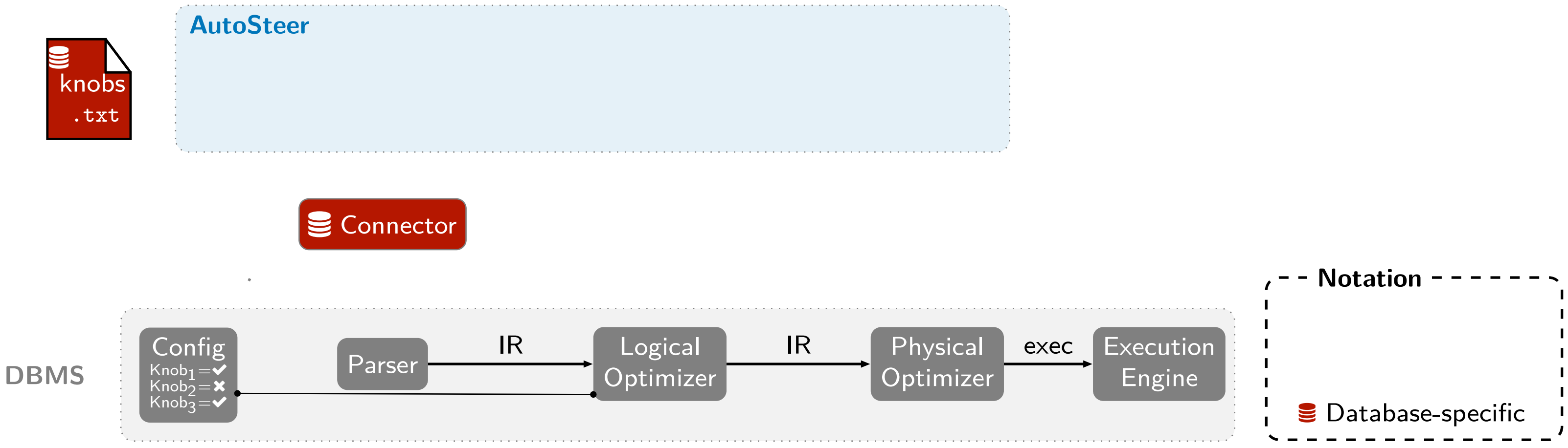
AutoSteer – Overview



AutoSteer – Overview



AutoSteer – Overview



AutoSteer - Generic DBMS Connector

Required Functionality:

- Set up database connection
- Toggle optimizer knobs
- Explain and analyze SQL queries
- Execute SQL queries
- Track execution times

AutoSteer - Generic DBMS Connector

Required Functionality:

- Set up database connection
- Toggle optimizer knobs
- Explain and analyze SQL queries
- Execute SQL queries
- Track execution times

```python
class DBConnector:
  def set_knob(knob: str, enable: bool) -> None:
    """Disable the provided list of knobs"""
    raise NotImplementedError()

  def get_knob(self, knob: str) -> bool:
    """Get current status of a knob"""
    raise NotImplementedError()

  def explain(self, query: str) -> str:
    """Explain a query and return its plan"""
    raise NotImplementedError()

  def execute(self, query: str) -> TimedResult:
    """Execute query and measure time"""
    raise NotImplementedError()
```

AutoSteer - PostgreSQL Connector

```python
class PostgreSQLConnector(DBConnector):
  def __init__(url: str):
    self.conn = ... # setup PostgreSQL connection

  def set_knob(knob: str, enable: bool) -> None:
    self.conn.exec(f"SET {knob} TO {'ON' if enable else 'OFF'}")

  def execute(query: str) -> dict:
    return self.conn.exec(query)

  def explain(query: str) -> dict:
    return self.execute(f'EXPLAIN (FORMAT JSON) {query}')
```

AutoSteer - PostgreSQL Connector

```python
class PostgreSQLConnector(DBConnector):
  def __init__(url: str):
    self.conn = ... # setup PostgreSQL connection

  def set_knob(knob: str, enable: bool) -> None:
    self.conn.exec(f"SET {knob} TO {'ON' if enable else 'OFF'}")

  def execute(query: str) -> dict:
    return self.conn.exec(query)

  def explain(query: str) -> dict:
    return self.execute(f'EXPLAIN (FORMAT JSON) {query}')
```

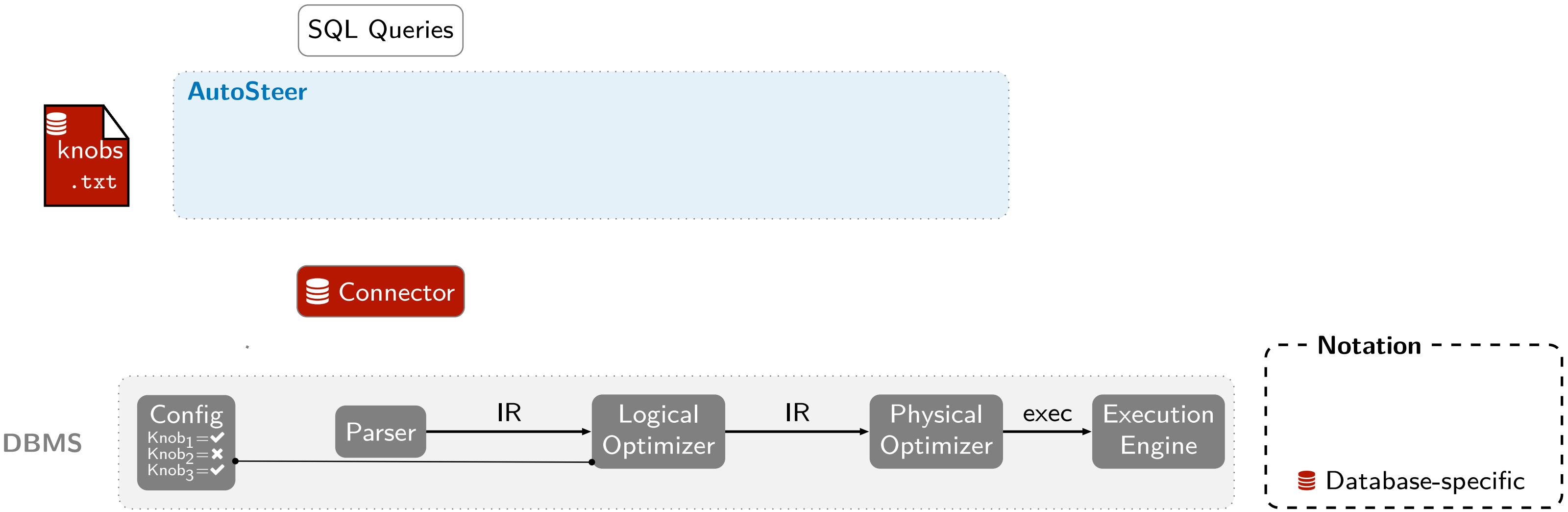
⇒ **DB Connectors mainly differ in the syntax to toggle knobs and to explain plans!**

AutoSteer - PostgreSQL Connector

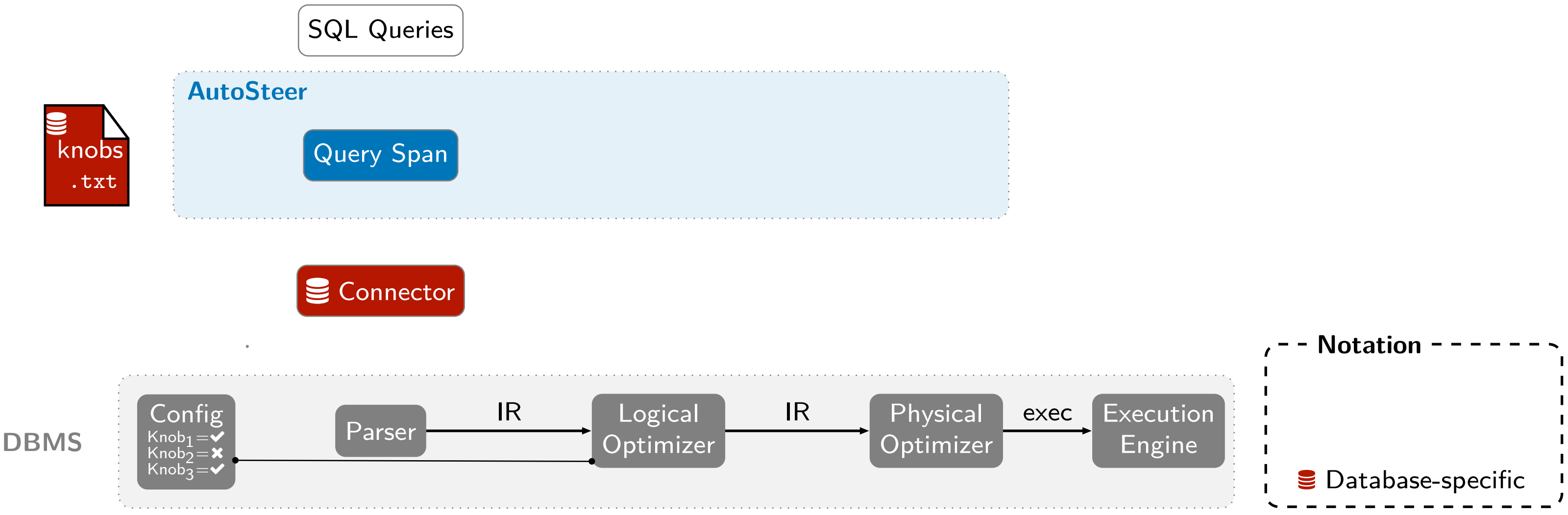
```python
class PostgreSQLConnector(DBConnector):
  def __init__(url: str):
    self.conn = ... # setup PostgreSQL connection

  def set_knob(knob: str, enable: bool) -> None:
    self.conn.exec(f"SET {knob} TO {'ON' if enable else 'OFF'}")

  def execute(query: str) -> dict:
    return self.conn.exec(query)

  def explain(query: str) -> dict:
    return self.execute(f'EXPLAIN (FORMAT JSON) {query}')
```

ℹ AutoSteer also supports **integrated database connectors (AutoSteer-C).**

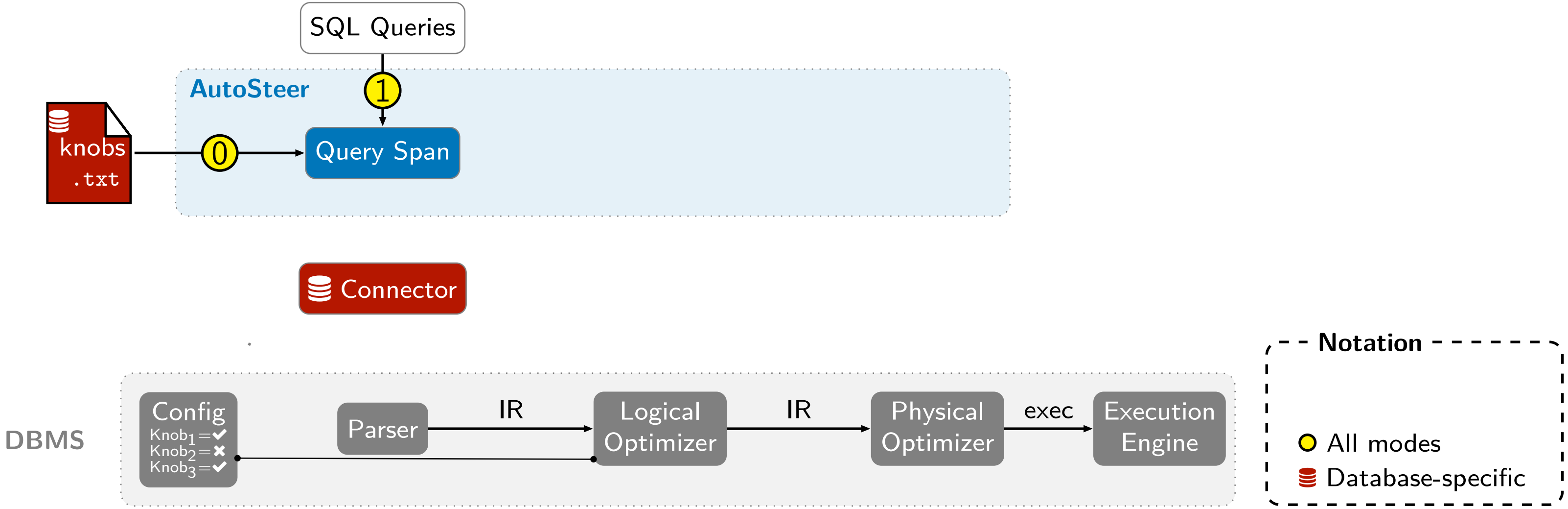
AutoSteer – Overview



AutoSteer – Overview



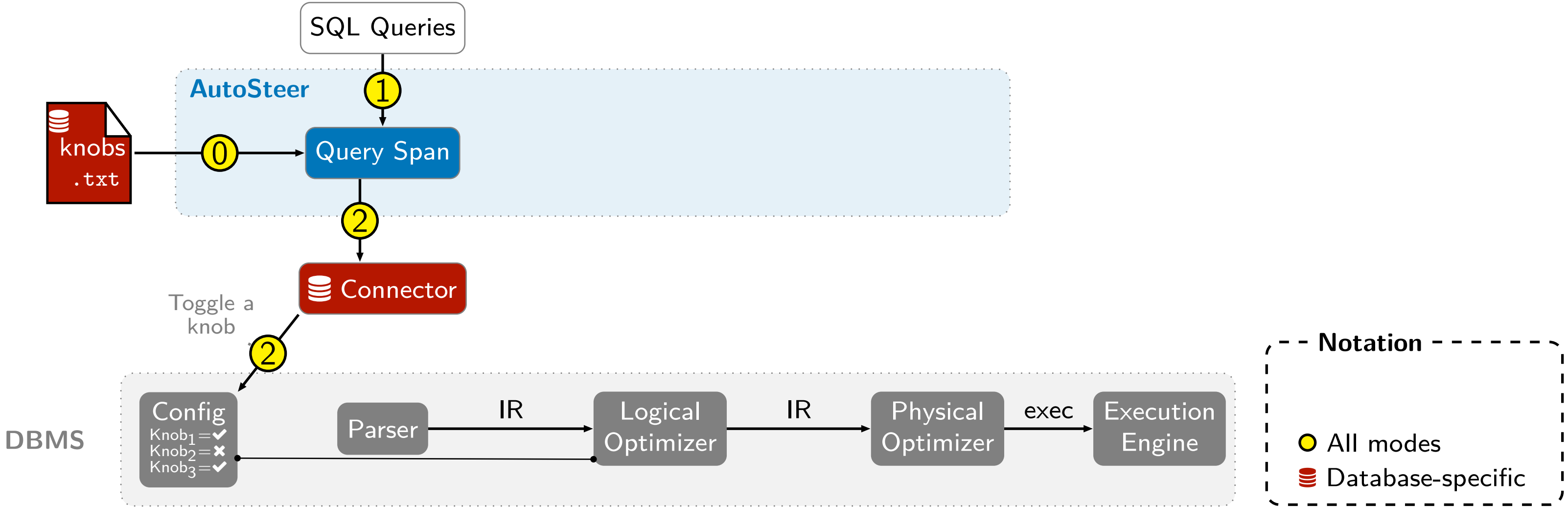
AutoSteer – Overview



Query Span

A query span contains all the knobs that contribute to the query plan's optimization!

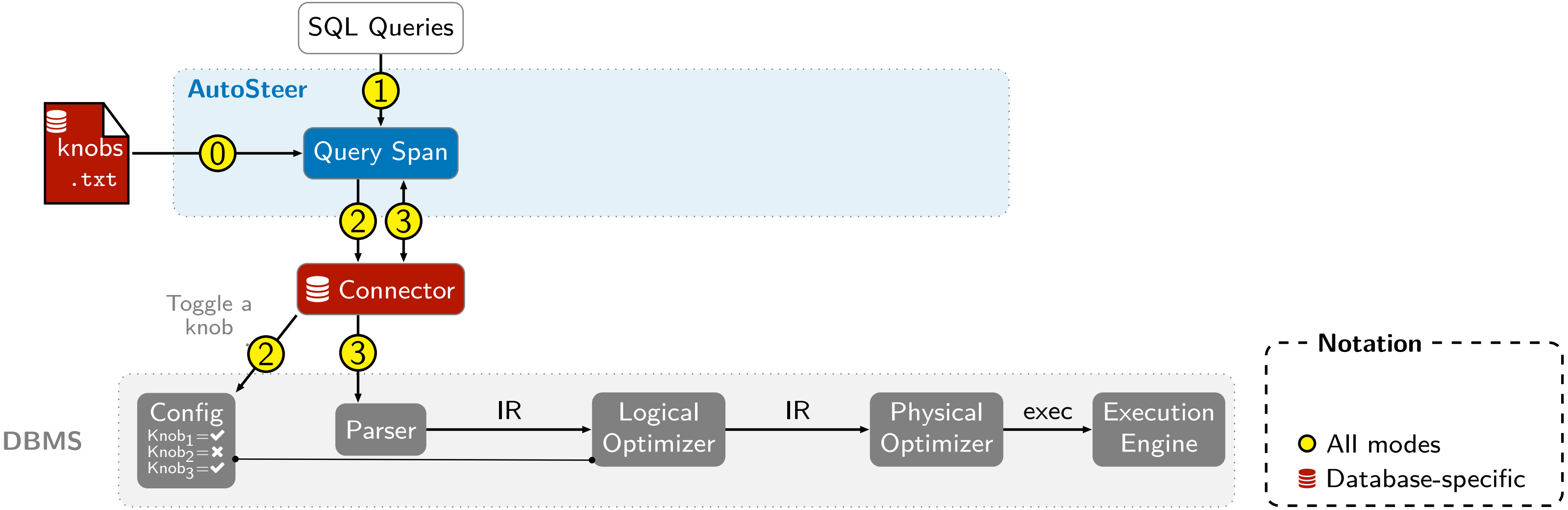
AutoSteer – Overview



Query Span

A query span contains all the knobs that contribute to the query plan's optimization!

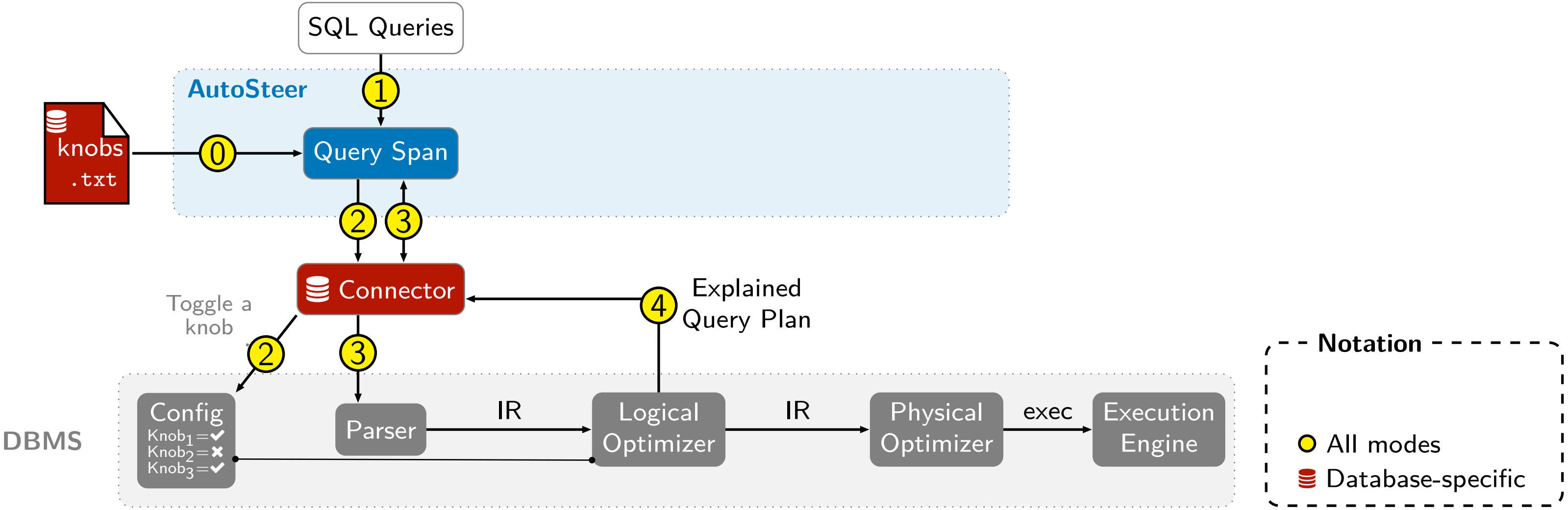
AutoSteer – Overview



Query Span

A query span contains all the knobs that contribute to the query plan's optimization!

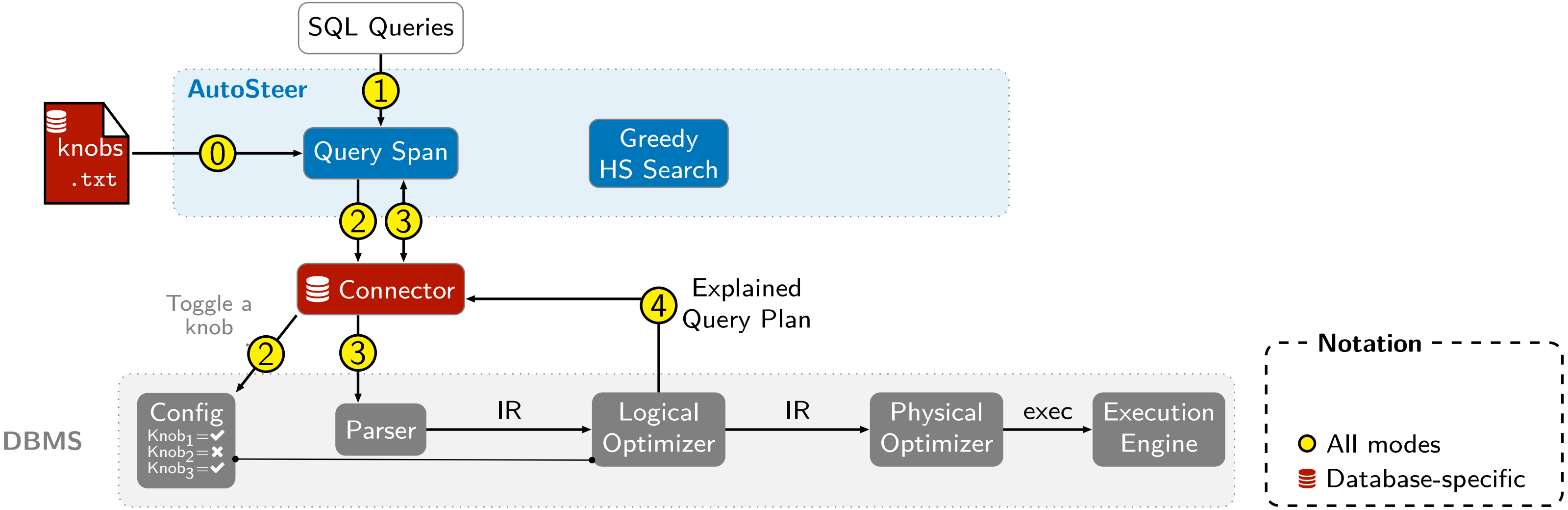
AutoSteer – Overview



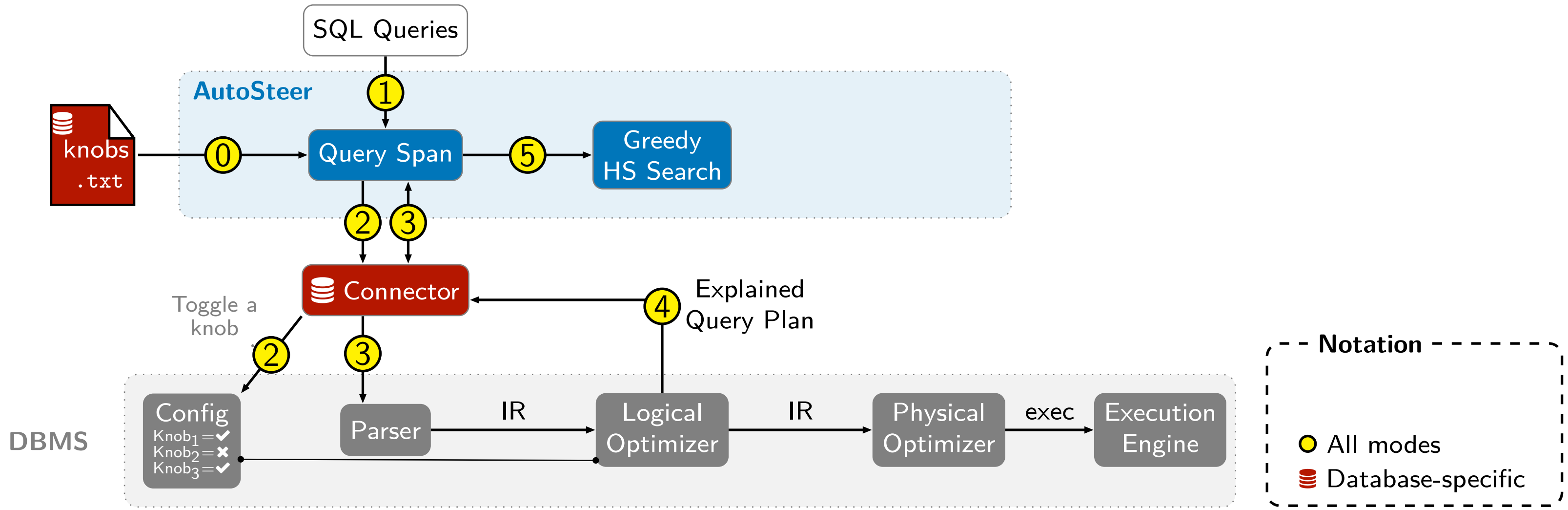
Query Span

A query span contains all the knobs that contribute to the query plan's optimization!

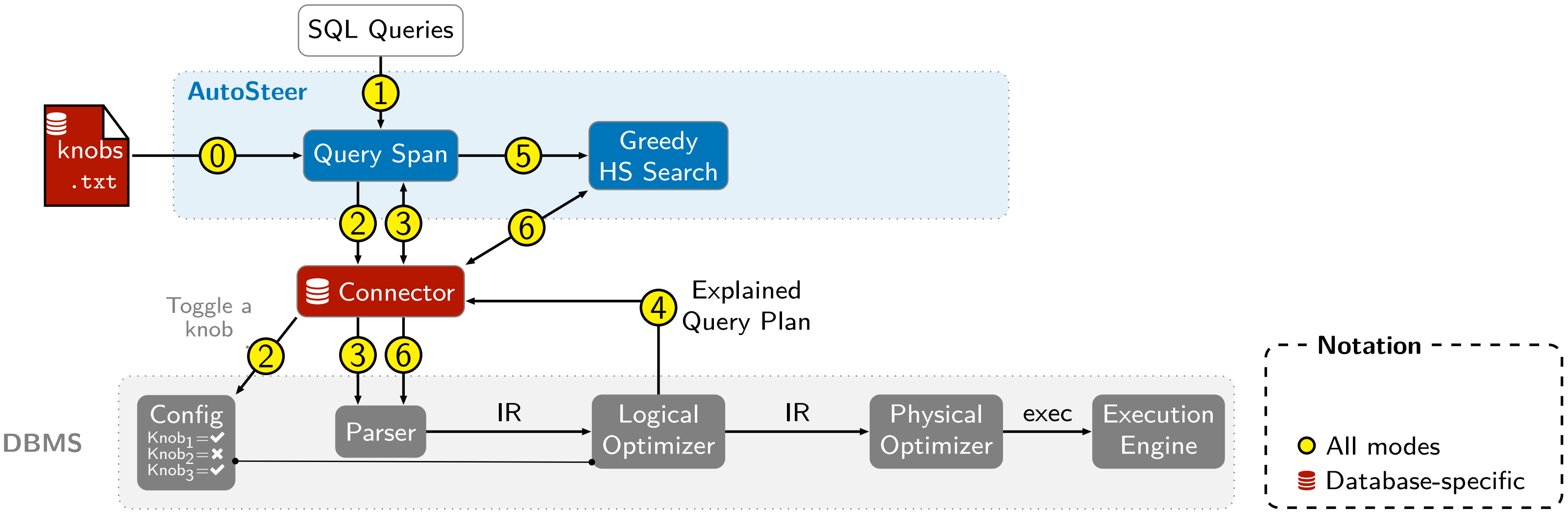
AutoSteer – Overview



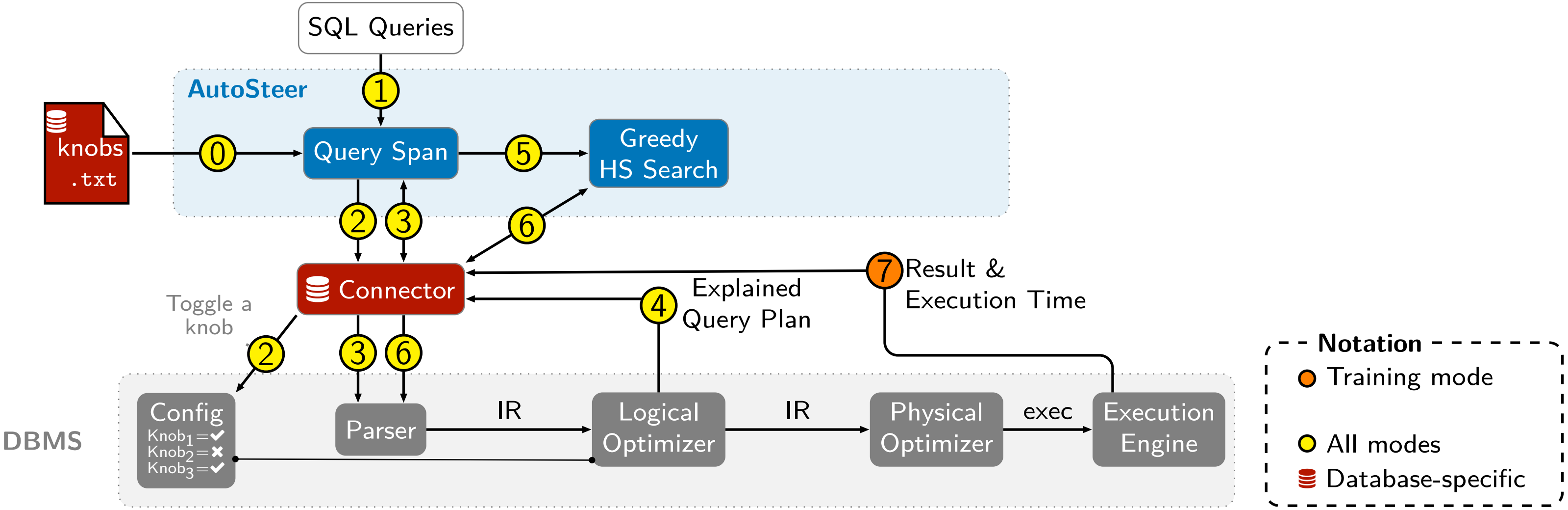
AutoSteer – Overview



AutoSteer – Overview



AutoSteer – Overview



Greedy Hint-Set Search

- There are $2^{|\text{Query Span}|}$ different hint-sets; However, most yield bad query plans.

Greedy Hint-Set Search

- There are $2^{|\text{Query Span}|}$ different hint-sets; However, most yield bad query plans.
- Greedy search generates promising hint-sets (HS) with reasonable overhead.

Greedy Hint-Set Search

- There are $2^{|\text{Query Span}|}$ different hint-sets; However, most yield bad query plans.
- Greedy search generates promising hint-sets (HS) with reasonable overhead.
  - **Assumption**: Larger, beneficial HSs consist of smaller, beneficial HSs
    Not always, but in many cases true – experimentally tested ✔

Greedy Hint-Set Search

- There are $2^{|\text{Query Span}|}$ different hint-sets; However, most yield bad query plans.
- Greedy search generates promising hint-sets (HS) with reasonable overhead.
  - **Assumption**: Larger, beneficial HSs consist of smaller, beneficial HSs
    Not always, but in many cases true – experimentally tested ✔
- **Input**: SQL Query and Query Span (example: $[k_1, k_3, k_6, k_9]$)

Greedy Hint-Set Search

- There are $2^{|\text{Query Span}|}$ different hint-sets; However, most yield bad query plans.
- Greedy search generates promising hint-sets (HS) with reasonable overhead.
  - **Assumption**: Larger, beneficial HSs consist of smaller, beneficial HSs
    Not always, but in many cases true – experimentally tested ✔
- **Input**: SQL Query and Query Span (example: $[k_1, k_3, k_6, k_9]$)

Default Plan: $\boxed{\varnothing \quad \rightarrow 60s}$

Greedy Hint-Set Search

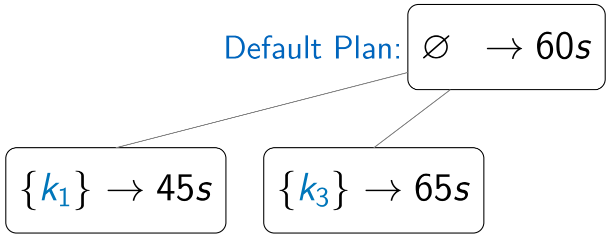
- There are $2^{|\text{Query Span}|}$ different hint-sets; However, most yield bad query plans.
- Greedy search generates promising hint-sets (HS) with reasonable overhead.
  - **Assumption**: Larger, beneficial HSs consist of smaller, beneficial HSs
    Not always, but in many cases true – experimentally tested ✔
- **Input**: SQL Query and Query Span (example: $[k_1, k_3, k_6, k_9]$)

Default Plan: $\varnothing \rightarrow 60s$

$\{k_1\} \rightarrow 45s$

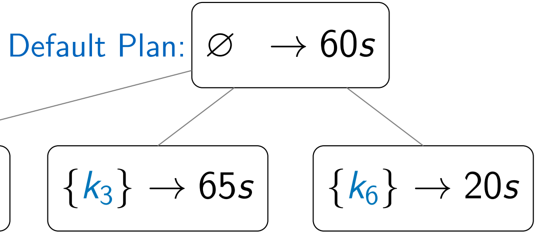
Greedy Hint-Set Search

- There are $2^{|\text{Query Span}|}$ different hint-sets; However, most yield bad query plans.
- Greedy search generates promising hint-sets (HS) with reasonable overhead.
  - **Assumption**: Larger, beneficial HSs consist of smaller, beneficial HSs
    Not always, but in many cases true – experimentally tested ✔
- **Input**: SQL Query and Query Span (example: $[k_1, k_3, k_6, k_9]$)



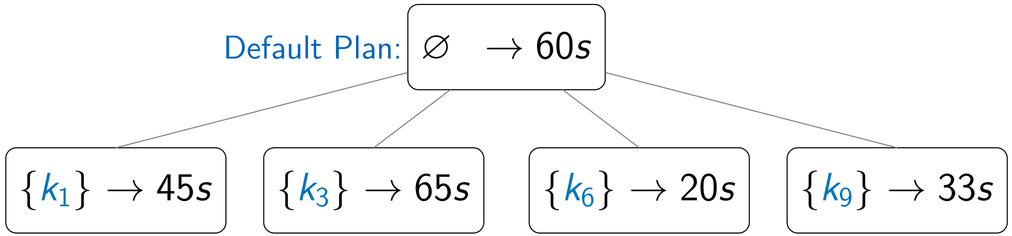
Greedy Hint-Set Search

- There are $2^{|\text{Query Span}|}$ different hint-sets; However, most yield bad query plans.
- Greedy search generates promising hint-sets (HS) with reasonable overhead.
  - **Assumption**: Larger, beneficial HSs consist of smaller, beneficial HSs
    Not always, but in many cases true – experimentally tested ✔
- **Input**: SQL Query and Query Span (example: $[k_1, k_3, k_6, k_9]$)



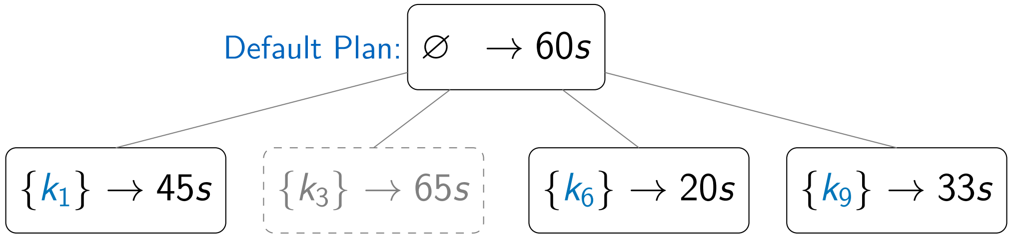
Greedy Hint-Set Search

- There are $2^{|\text{Query Span}|}$ different hint-sets; However, most yield bad query plans.
- Greedy search generates promising hint-sets (HS) with reasonable overhead.
  - **Assumption**: Larger, beneficial HSs consist of smaller, beneficial HSs
    Not always, but in many cases true – experimentally tested ✔
- **Input**: SQL Query and Query Span (example: $[k_1, k_3, k_6, k_9]$)



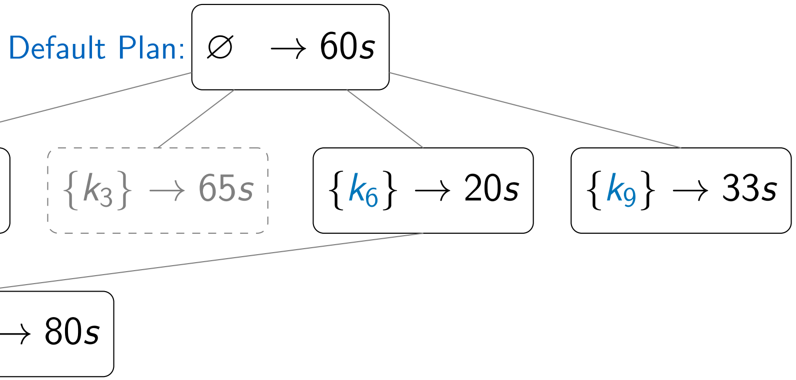
Greedy Hint-Set Search

- There are $2^{|\text{Query Span}|}$ different hint-sets; However, most yield bad query plans.
- Greedy search generates promising hint-sets (HS) with reasonable overhead.
  - **Assumption**: Larger, beneficial HSs consist of smaller, beneficial HSs
    Not always, but in many cases true – experimentally tested ✔
- **Input**: SQL Query and Query Span (example: $[k_1, k_3, k_6, k_9]$)



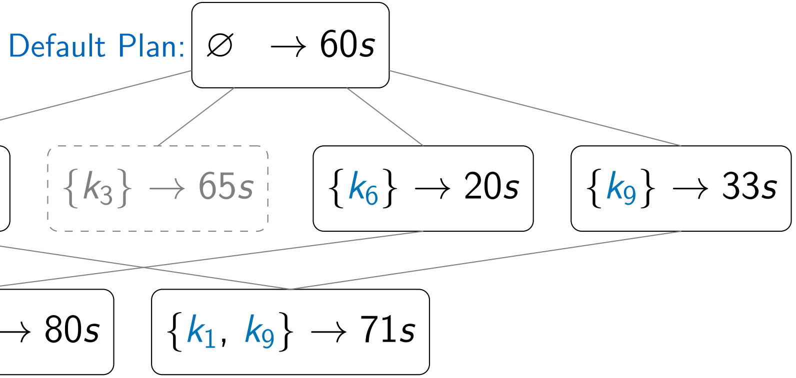
Greedy Hint-Set Search

- There are $2^{|\text{Query Span}|}$ different hint-sets; However, most yield bad query plans.
- Greedy search generates promising hint-sets (HS) with reasonable overhead.
  - **Assumption**: Larger, beneficial HSs consist of smaller, beneficial HSs
    Not always, but in many cases true – experimentally tested ✔
- **Input**: SQL Query and Query Span (example: $[k_1, k_3, k_6, k_9]$)



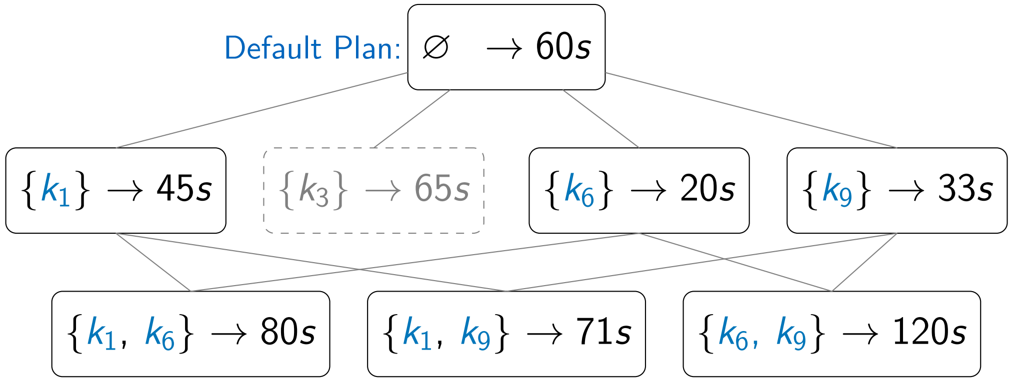
Greedy Hint-Set Search

- There are $2^{|\text{Query Span}|}$ different hint-sets; However, most yield bad query plans.
- Greedy search generates promising hint-sets (HS) with reasonable overhead.
  - **Assumption**: Larger, beneficial HSs consist of smaller, beneficial HSs
    Not always, but in many cases true – experimentally tested ✔
- **Input**: SQL Query and Query Span (example: $[k_1, k_3, k_6, k_9]$)



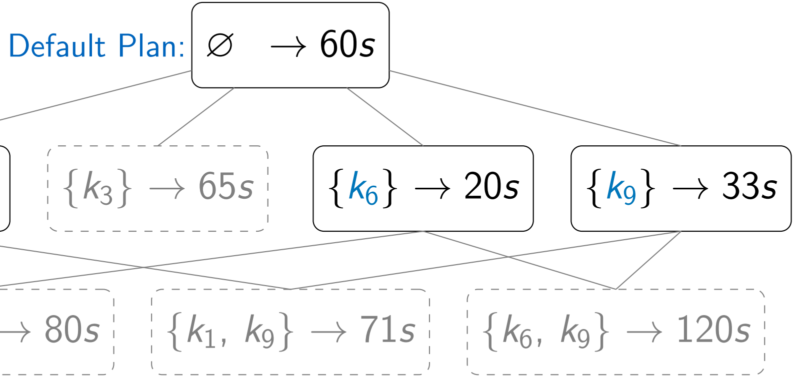
Greedy Hint-Set Search

- There are $2^{|\text{Query Span}|}$ different hint-sets; However, most yield bad query plans.
- Greedy search generates promising hint-sets (HS) with reasonable overhead.
  - **Assumption**: Larger, beneficial HSs consist of smaller, beneficial HSs
    Not always, but in many cases true – experimentally tested ✔
- **Input**: SQL Query and Query Span (example: $[k_1, k_3, k_6, k_9]$)

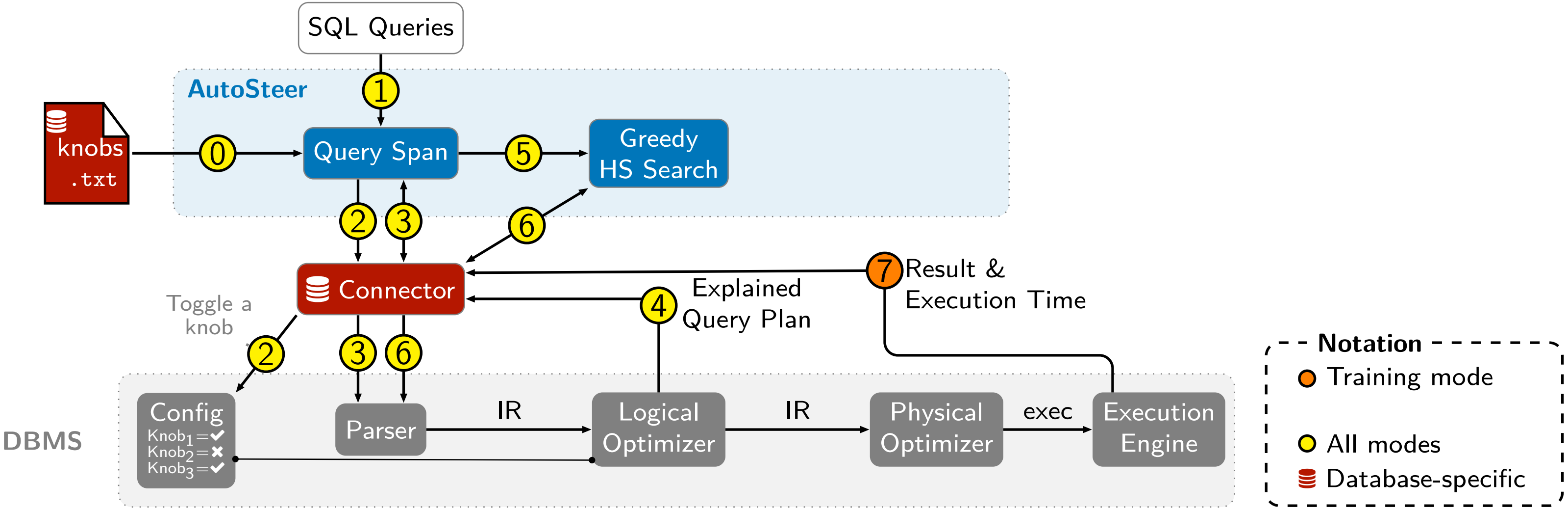


Greedy Hint-Set Search

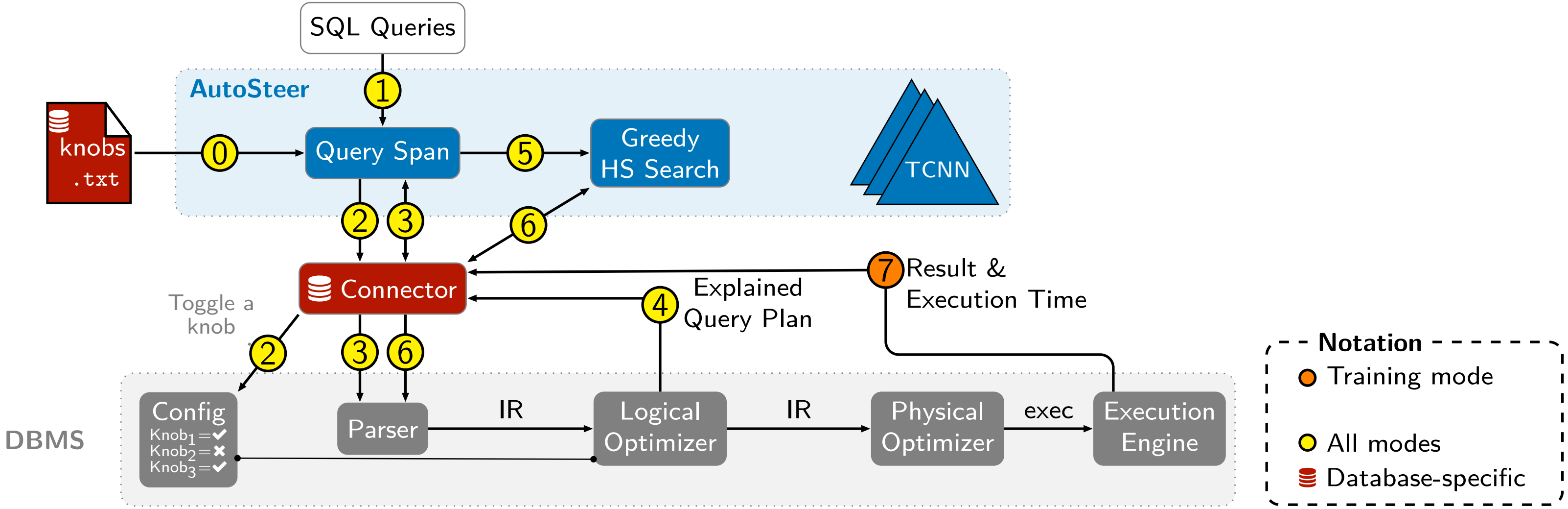
- There are $2^{|\text{Query Span}|}$ different hint-sets; However, most yield bad query plans.
- Greedy search generates promising hint-sets (HS) with reasonable overhead.
  - **Assumption**: Larger, beneficial HSs consist of smaller, beneficial HSs
    Not always, but in many cases true – experimentally tested ✔
- **Input**: SQL Query and Query Span (example: $[k_1, k_3, k_6, k_9]$)



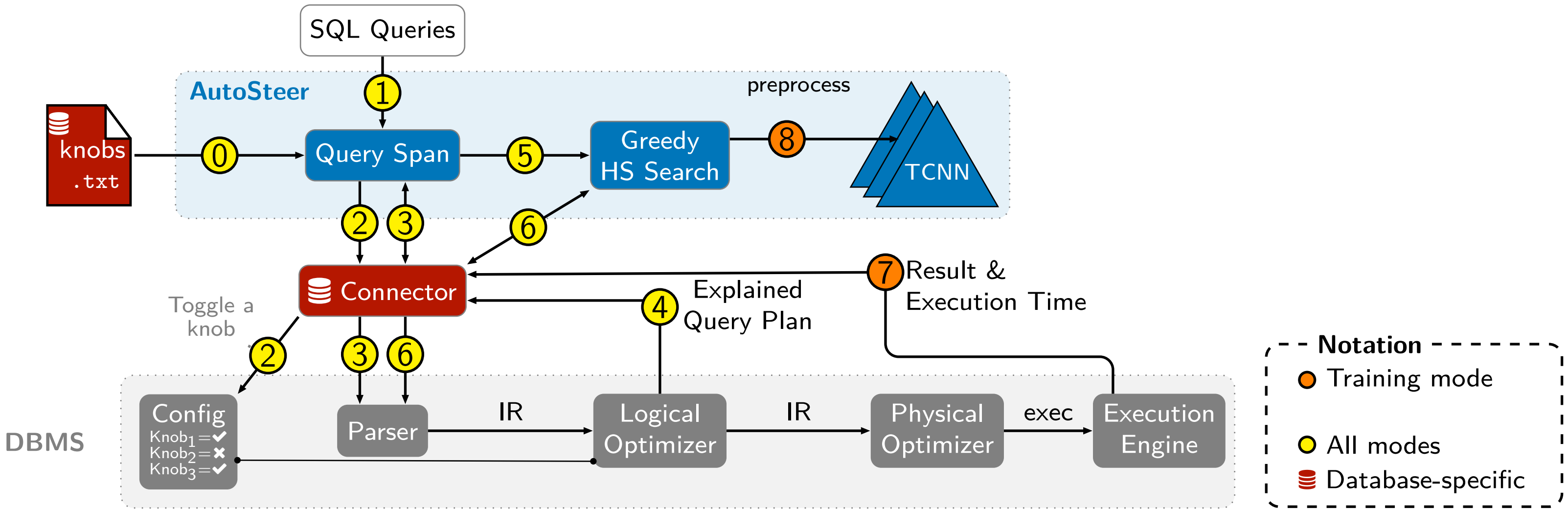
AutoSteer – Overview



AutoSteer – Overview



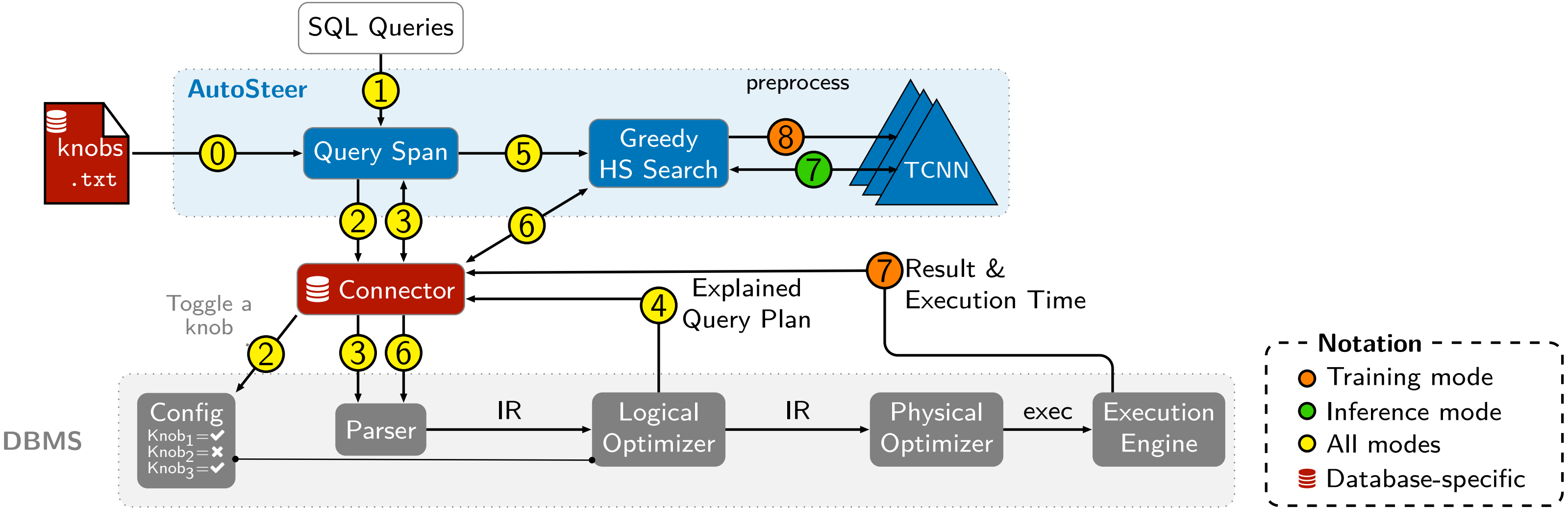
AutoSteer – Overview



Training Mode

AutoSteer **generates training data** by exploring and **executing** alternative plans.

AutoSteer – Overview



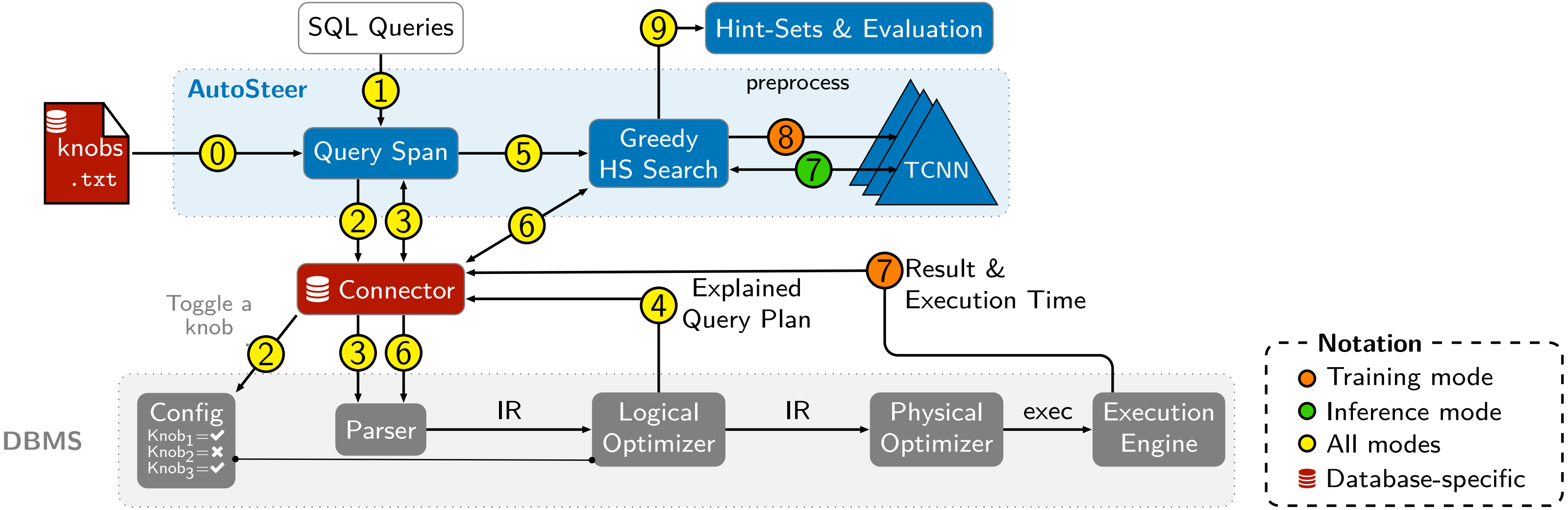
Training Mode

AutoSteer **generates training data** by exploring and **executing** alternative plans.

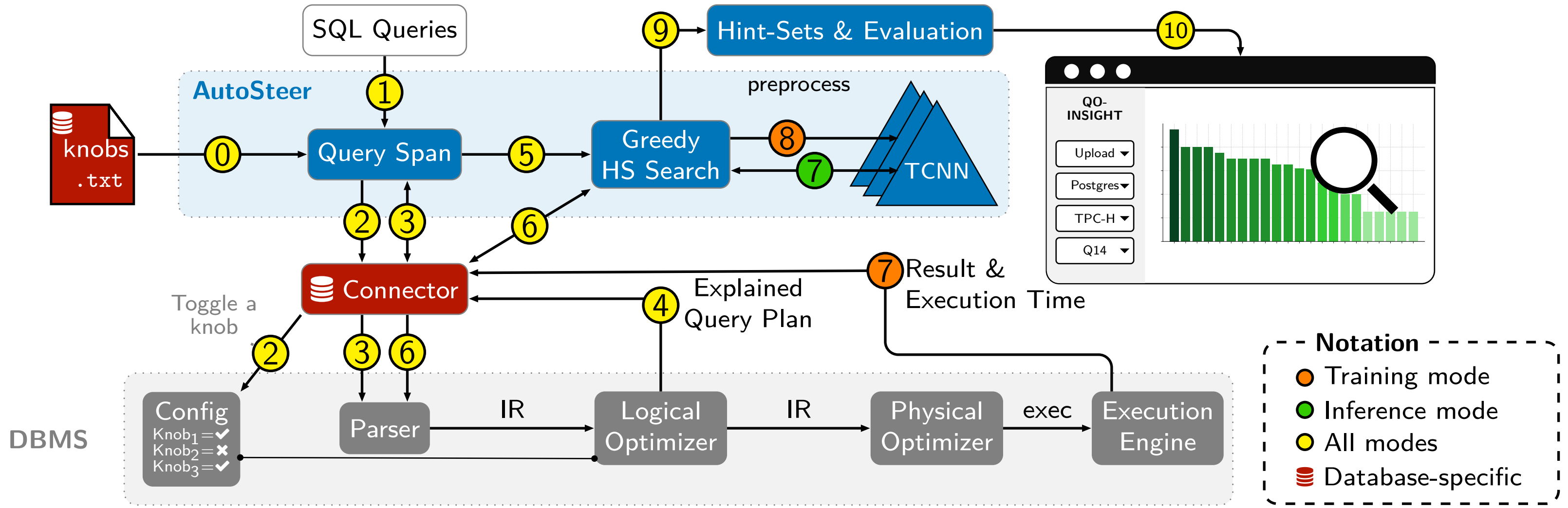
Inference Mode

AutoSteer **steers queries at runtime** and uses the TCNN to **infer execution times**.

AutoSteer – Overview



AutoSteer – Overview



VLDB 2023: "QO-Insight: Inspecting Steered Query Optimizers" → Demo Group B

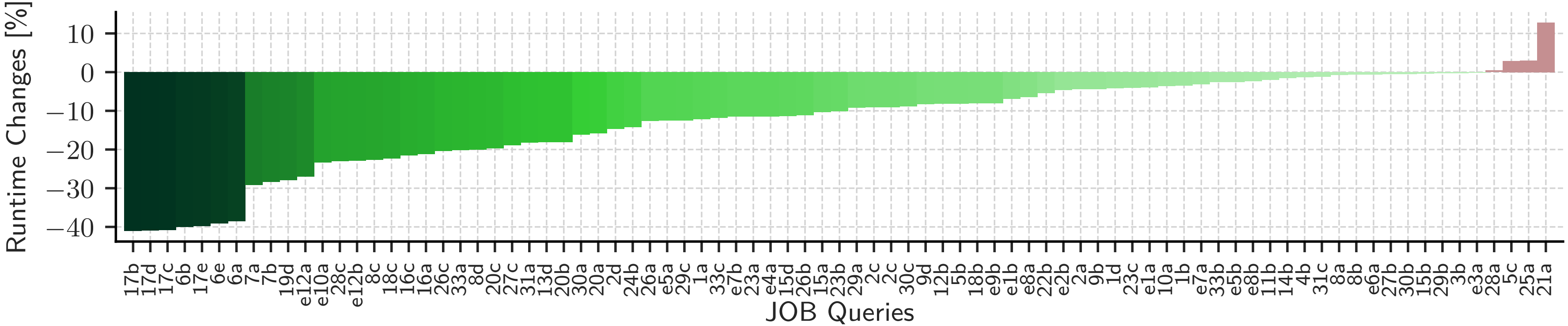
Evaluation

Evaluation of AutoSteer

- Tested AutoSteer with **PostgreSQL**, **DuckDB**, **PrestoDB**, **MySQL**, and **SparkSQL**
- Evaluation based on **public benchmarks** and **production workloads at Meta**
  - Join Order Benchmark (137 queries)
  - TPC-DS (100 queries)
  - Stackexchange (100 queries)
  - Dashboard application at Meta (>3000 queries, scanning PBs of data)
- Comparison to **previous steering approaches** (cf. to the paper for more details!)

Join Order Benchmark – PrestoDB

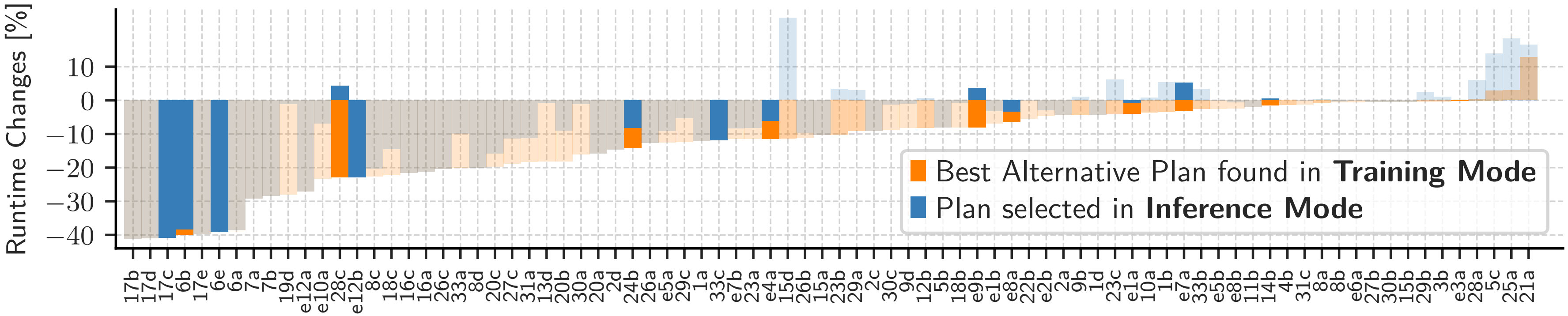
- For 137 queries, AutoSteer-C (w/ integrated connector) explored 1730 different hint-sets
- Evaluated between 8 and 34 different plans per query
- Achieved improvements of **up to 40%**



Relative performance changes of the best known alternative plan compared to the default plan.

Join Order Benchmark – PrestoDB

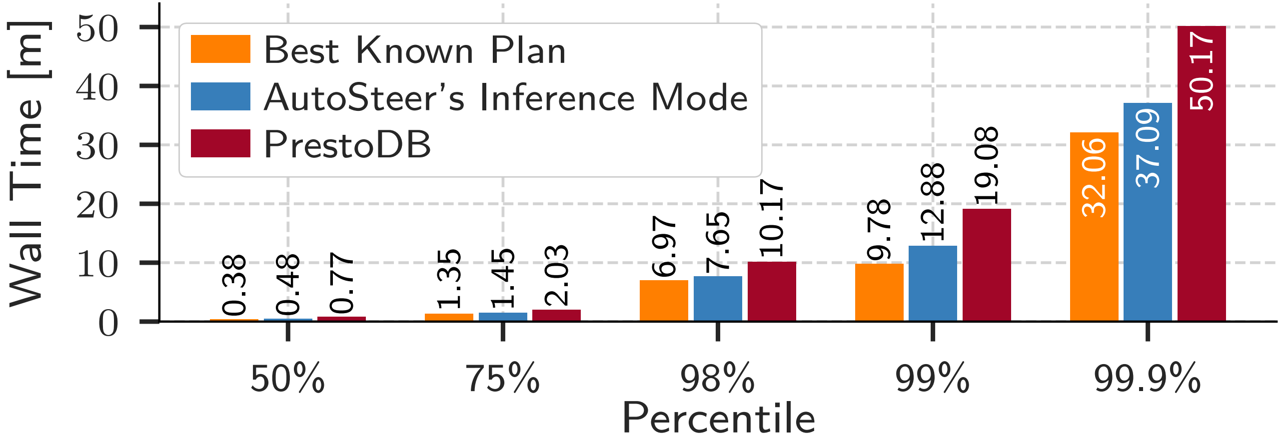
- AutoSteer-C (w/ integrated connector) using a tree convolutional neural network to infer execution times
- Reduces execution times of
  - unseen queries by 20.6% (opaque)
  - seen queries by 26.8% (transparent)



Relative performance changes of the optimal and the selected plan wrt. the default plan.

Dashboard Application at Meta – PrestoDB

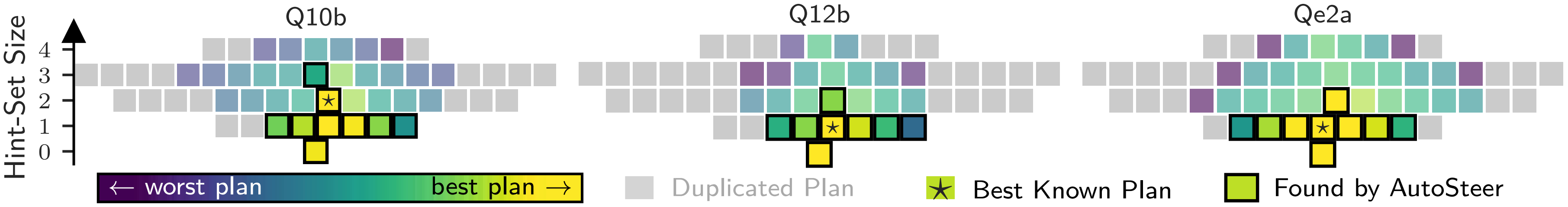
- Focus on **tail latencies**
- >3000 Queries, scanning PBs of data, hundreds of worker nodes
- Workload runs multiple times per day



AutoSteer significantly reduces tail latencies of production workloads at Meta

Greedy Exploration

- **Assumption**: Larger, beneficial HSs consist of smaller, beneficial HSs
- Greedy explores significantly fewer hint-sets than previous approaches (like Bao),
- but it meets or outperforms their performance improvements



Conclusions

- AutoSteer is a practical framework to
  - steer query optimizers at runtime and to
  - find counter examples to debug and further improve existing query optimizers

Conclusions

- AutoSteer is a practical framework to
  - steer query optimizers at runtime and to
  - find counter examples to debug and further improve existing query optimizers
- AutoSteer **outperforms previous steering approaches** wrt.
  - exploration time, achieved performance improvements, and applicability to new systems

Conclusions

- AutoSteer is a practical framework to
  - steer query optimizers at runtime and to
  - find counter examples to debug and further improve existing query optimizers

- AutoSteer **outperforms previous steering approaches** wrt.
  - exploration time, achieved performance improvements, and applicability to new systems

- AutoSteer is open-source and already supports five DBMSs (`https://github.com/IntelLabs/Auto-Steer`)

Conclusions

- AutoSteer is a practical framework to
  - steer query optimizers at runtime and to
  - find counter examples to debug and further improve existing query optimizers
- AutoSteer **outperforms previous steering approaches** wrt.
  - exploration time, achieved performance improvements, and applicability to new systems
- AutoSteer is open-source and already supports five DBMSs (`https://github.com/IntelLabs/Auto-Steer`)
- Applied AutoSteer in an industrial setting – lessons learned in the paper!

Conclusions

- AutoSteer is a practical framework to
  - steer query optimizers at runtime and to
  - find counter examples to debug and further improve existing query optimizers
- AutoSteer **outperforms previous steering approaches** wrt.
  - exploration time, achieved performance improvements, and applicability to new systems
- AutoSteer is open-source and already supports five DBMSs (`https://github.com/IntelLabs/Auto-Steer`)
- Applied AutoSteer in an industrial setting – lessons learned in the paper!

Thank you for your attention!