The 17th IEEE International Conference on
Trust, Security and Privacy in Computing and Communications
(IEEE TrustCom 2018)

# Query Guard: Privacy-preserving Latency-aware Query Optimization for Edge Computing


**Runhua Xu,** Balaji Palanisamy and James Joshi

University of Pittsburgh

Pittsburgh, PA, US

*runhua.xu@pitt.edu*


University of Pittsburgh
School of Computing and Information

LERSAIS
The Laboratory for Education and Research on Security Assured Information Systems

# Edge Computing

**It allows data produced by IoT devices to be processed geographically closer to where it is created instead of sending it across long routes to data centers/clouds.**

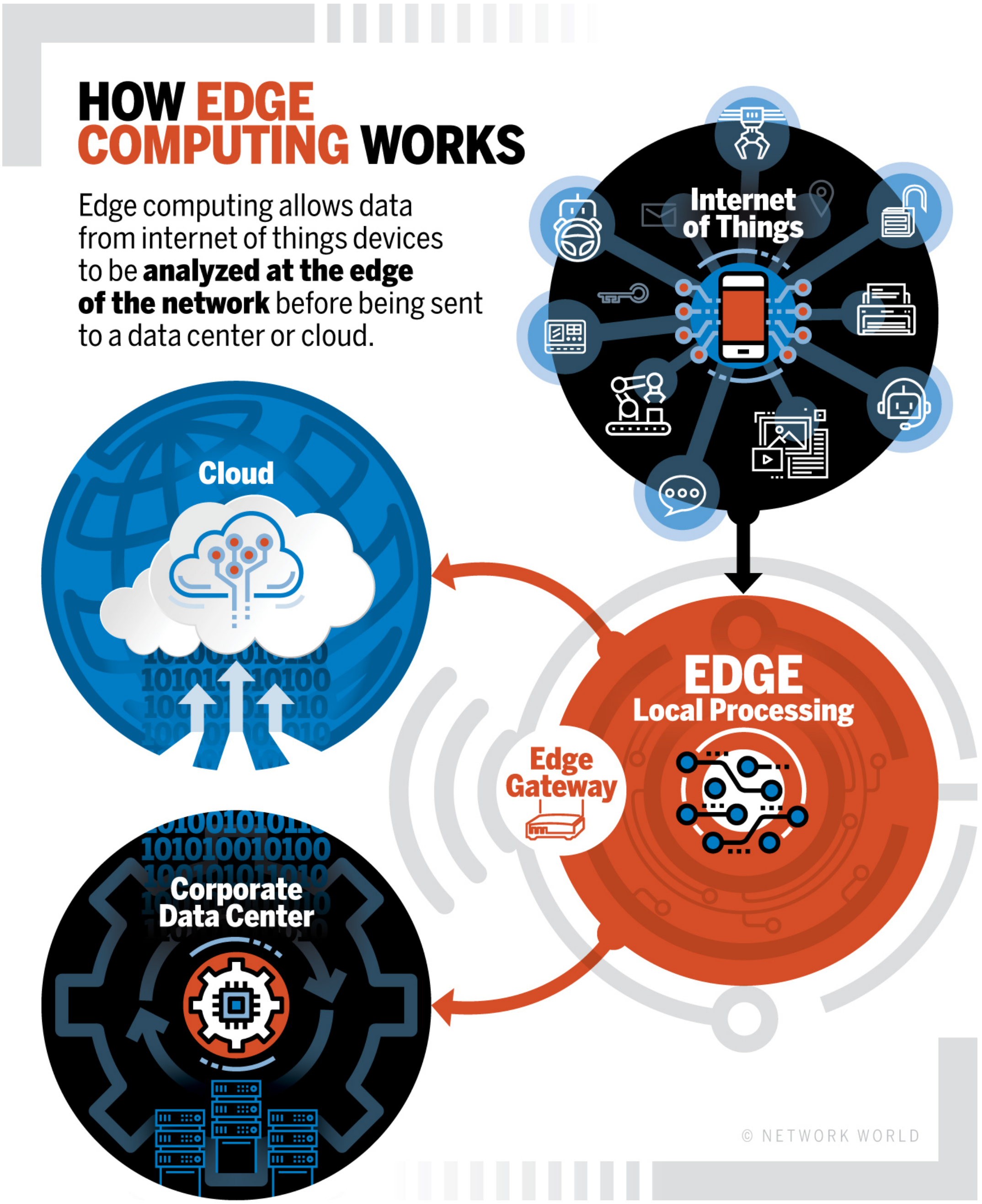
*(The background image is from xtelesis.com)*

The Laboratory for Education and Research on
Security Assured Information Systems (LERSAIS)

Wednesday, August 1, 2018

# Why/How Edge Computing

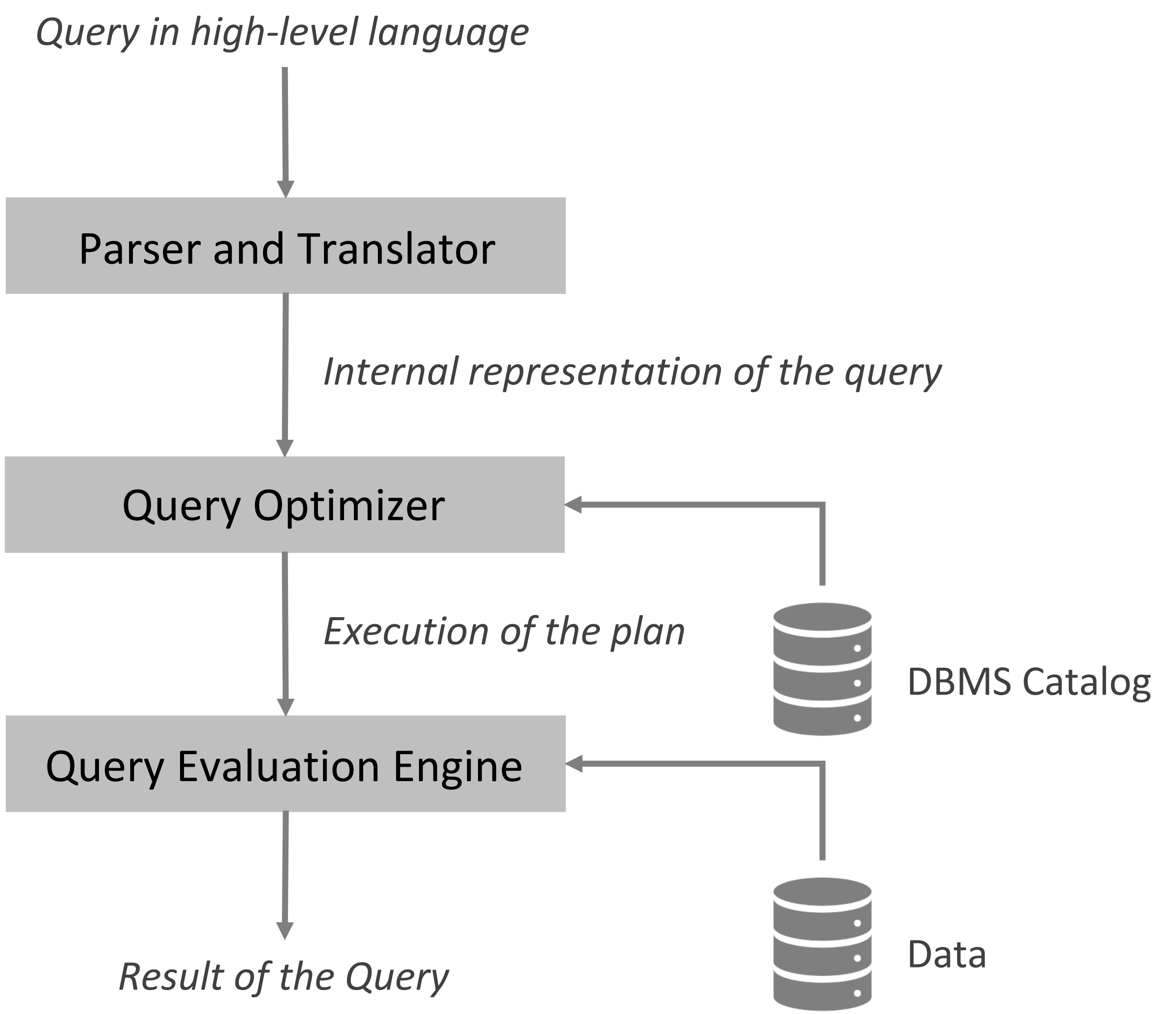
- Why does edge computing matter
  - *IoT devices have poor connectivity*
  - *It's not efficient for IoT devices to be constantly connected to a central cloud.*
  - *latency-sensitive processing requirement*
- How edge computing works
  - *Triage the data locally*



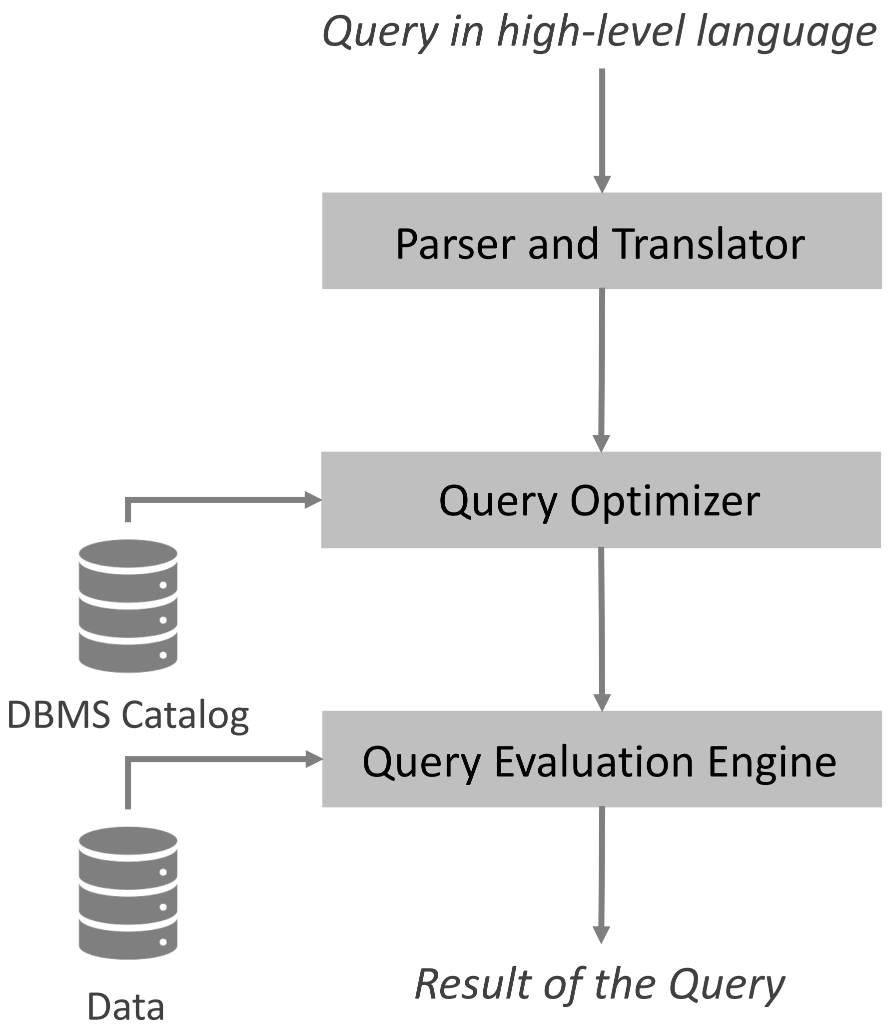
# Overview of Query Processing

**A 3-step Process**

*It transforms a high-level query into an equivalent and more efficient lower-level query.*



# Query Processing Example



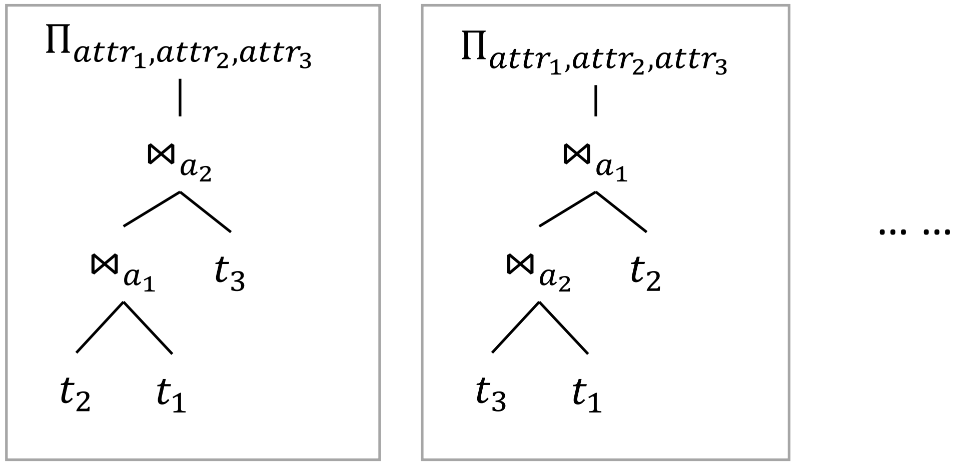

*SQL expression*

| | |
|---|---|
| SELECT | $attr_1, attr_2, attr_3$ |
| FROM | $t_1, t_2, t_3$ |
| WHERE | $t_1.a_1 = t_2.a_1$ AND $t_1.a_2 = t_3.a_2$ |

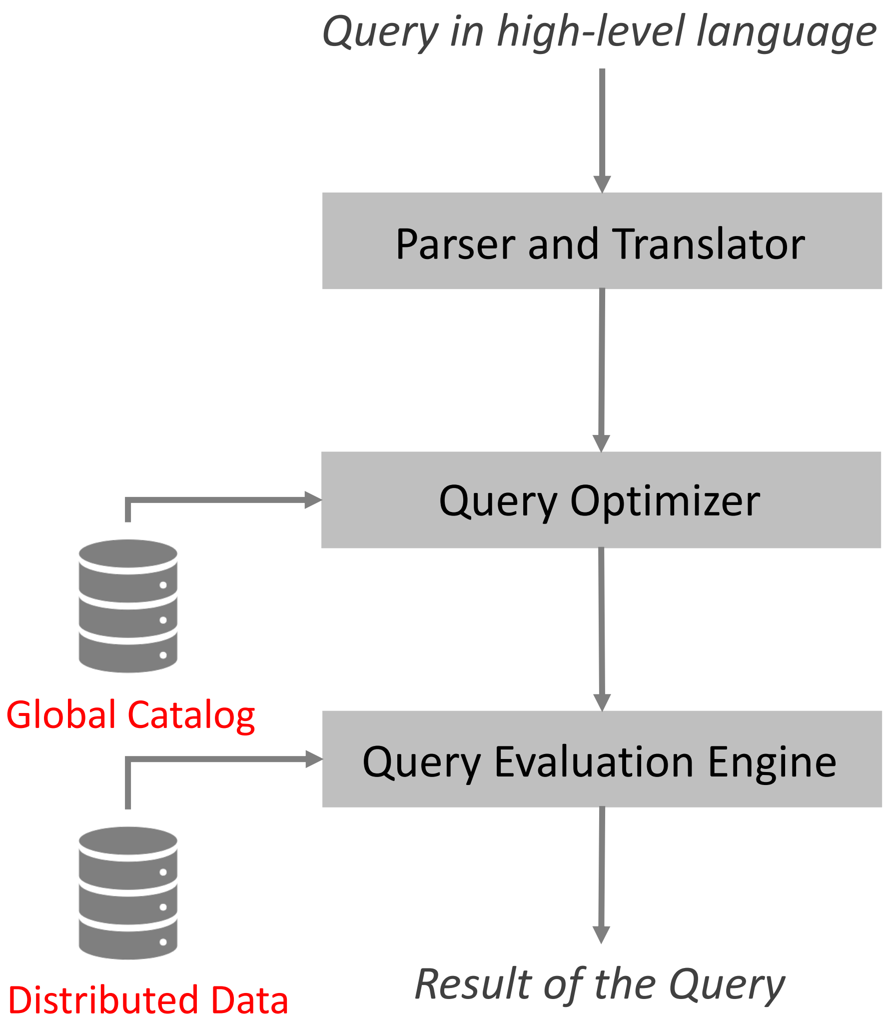
*Algebra expression*

$$\Pi_{attr_1,attr_2,attr_3}(t_2 \bowtie_{s_1} t_1 \bowtie_{s_2} t_3)$$

*Query execution plans*



# Distributed Query Processing Example




*SQL expression*

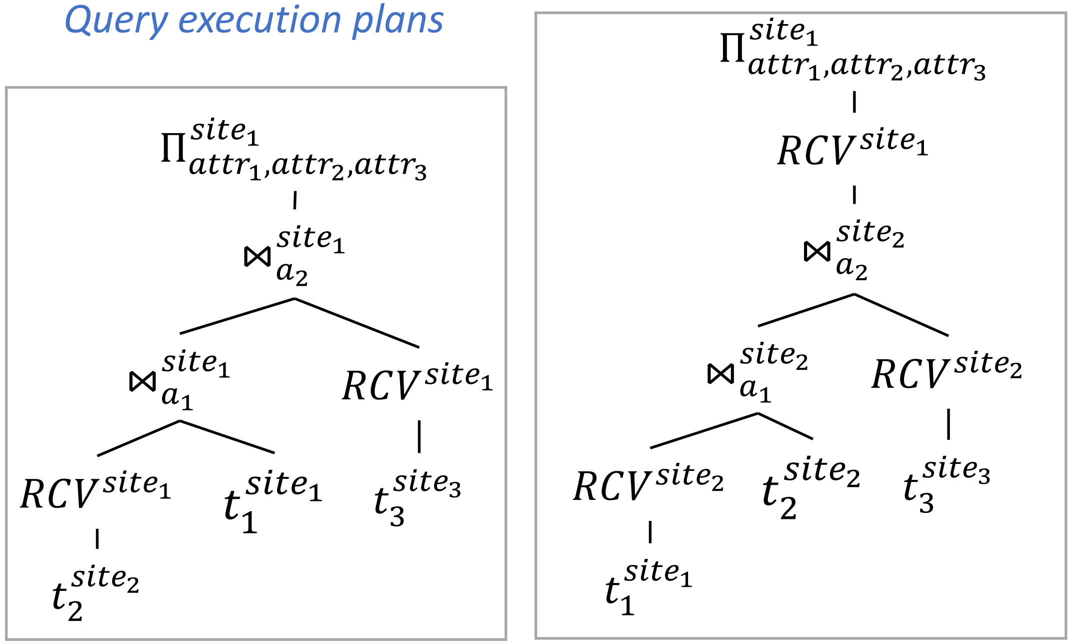
SELECT $attr_1$, $attr_2$, $attr_3$
FROM $t_1$, $t_2$, $t_3$
WHERE $t_1.a_1 = t_2.a_1$ AND $t_1.a_2 = t_3.a_2$

$t_1 \rightarrow site_1$
$t_2 \rightarrow site_2$
$t_3 \rightarrow site_3$

*Algebra expression*

$$\Pi_{attr_1,attr_2,attr_3}(t_2 \bowtie_{s_1} t_1 \bowtie_{s_2} t_3)$$

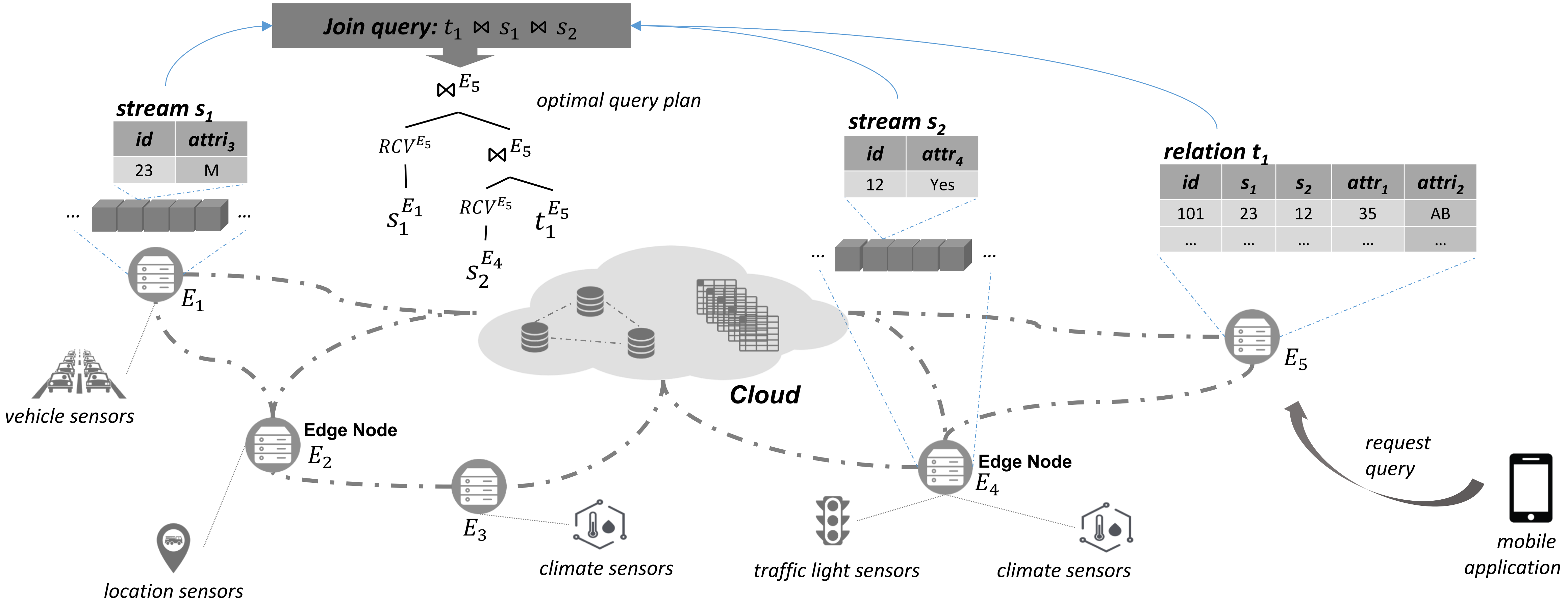
*Query execution plans*




Global Optimization
+
Local Optimization

… …

# Query Processing in Edge Computing




**SELECT** $attr_1, attr_2, attr_3, attr_4$
**FROM** $t_1$, $s_1$[RANGE 30], $s_2$[RANGE 30]
**WHERE** $t_1.s_1 = s_1.id$ **AND** $t_1.s_2 = s_2.id$

# Challenges and Concerns : Edge vs Cloud

- Management policy
  - *Cloud servers are managed through strict and regularized policies*
  - *Edge nodes may not have the same degree of regulatory and monitoring oversight.*
    - Ship selected/projected data to edge nodes that may be untrusted/semi-trusted
    - Lead to disclosure of private information within the edge nodes.
- Latency
  - *Cloud: query in/cross data center(s) → proprietary network bandwidth*
    - Emphasis of QO is primarily on minimizing the query computation time
  - *Edge: nodes are scattered geographically with varying degrees of network connectivity*
    - A special emphasis of QO is network latency or statbility

# Latency Analysis

Suppose that
edge nodes are located in city A
closest cloud data center is located in city B

**Edge-based approach**

$$t_{edge} = \max_{i \in \{e_2, e_3\}, j \in \{(e_1,e_2),(e_1,e_3)\}} (v_i t / v_{net} + t_j) + \mathcal{T}$$

22.223 ms

**Cloud-based approach**

$$t_{cloud} = \max_{i \in \{e_1, e_2, e_3\}} (v_i t / v_{net} + t_{a,b}) + \mathcal{T} + \sum v_i t / v_{net} + t_{a,b}$$

84.818 ms

TABLE I
SIMULATED LOCATIONS AND THEIR NETWORK LATENCY

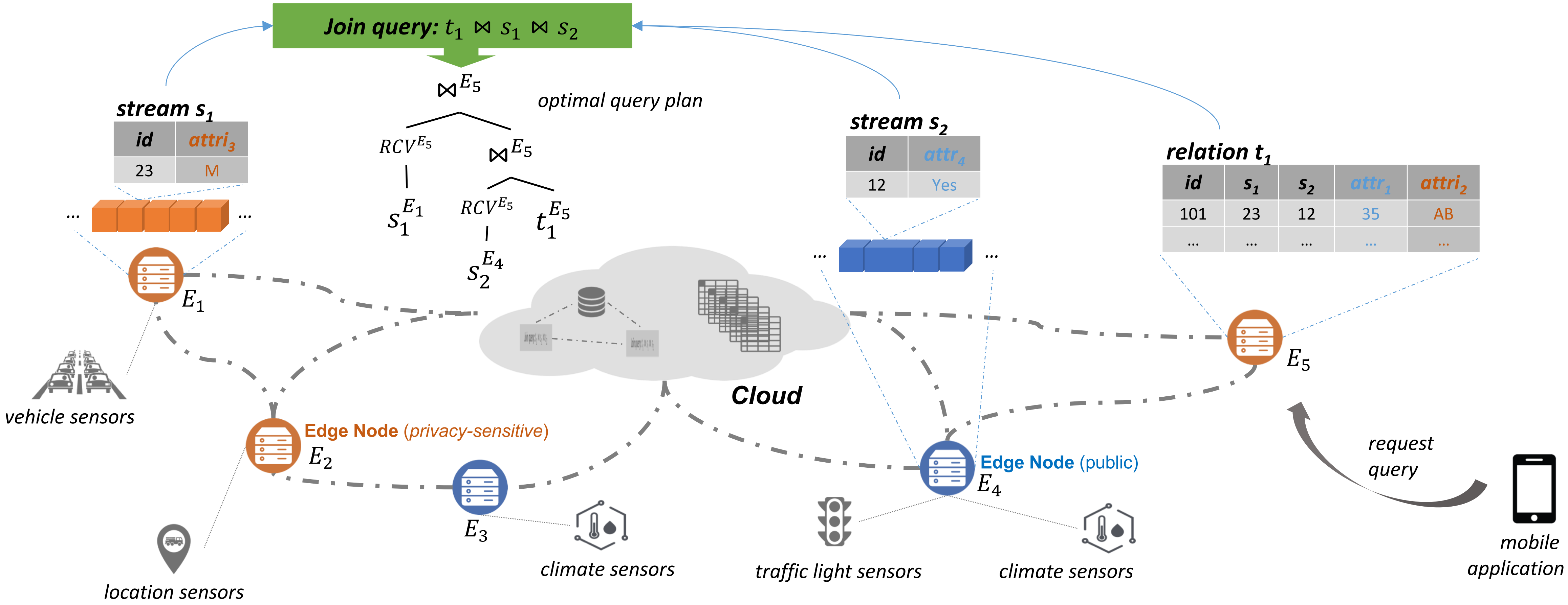
| Location | Distance | Latency(ms) | Result |
|---|---|---|---|
| $E_1$ - $E_2$ | $d_{e_1,e_2}$ | $0.022 \cdot d_{e_1,e_2} + 4.862$ | $t_{e_1,e_2}$ =5.082 |
| $E_2$ - $E_3$ | $d_{e_2,e_3}$ | $0.022 \cdot d_{e_2,e_3} + 4.862$ | $t_{e_2,e_3}$ =5.202 |
| $E_3$ - $E_1$ | $d_{e_3,e_1}$ | $0.022 \cdot d_{e_3,e_1} + 4.862$ | $t_{e_3,e_1}$ =5.192 |
| $A$ - $B$ | $d_{a,b}$ | $0.022 \cdot d_{a,b} + 4.862$ | $t_{a,b}$ =26.862 |

† Note that the latency of network traffic is estimated based on the distance using a linear model: $y = 0.022x + 4.862$ with coefficient of determination ($R^2 = 0.907$) proposed in [14].
‡ The distance between the data center and the city is assumed to be 1000 miles, while the distance between edge nodes is 10, 20, and 15 miles, respectively.

TABLE II
SIMULATED PARAMETER SETTINGS AND VALUES

| Symbol | Value | Description |
|---|---|---|
| $t$ | 30 min | Time interval of the query |
| $v_{e_1}$ | 1 KB/min | Speed of stream data generating at edge $E_1$ |
| $v_{e_2}$ | 2 KB/min | Speed of stream data generating at edge $E_2$ |
| $v_{e_3}$ | 3 KB/min | Speed of stream data generating at edge $E_3$ |
| $v_{net}$ | 100 Mbit/s | Ethernet speed |
| $\mathcal{T}$ | 10 ms | Query time in a single machine |

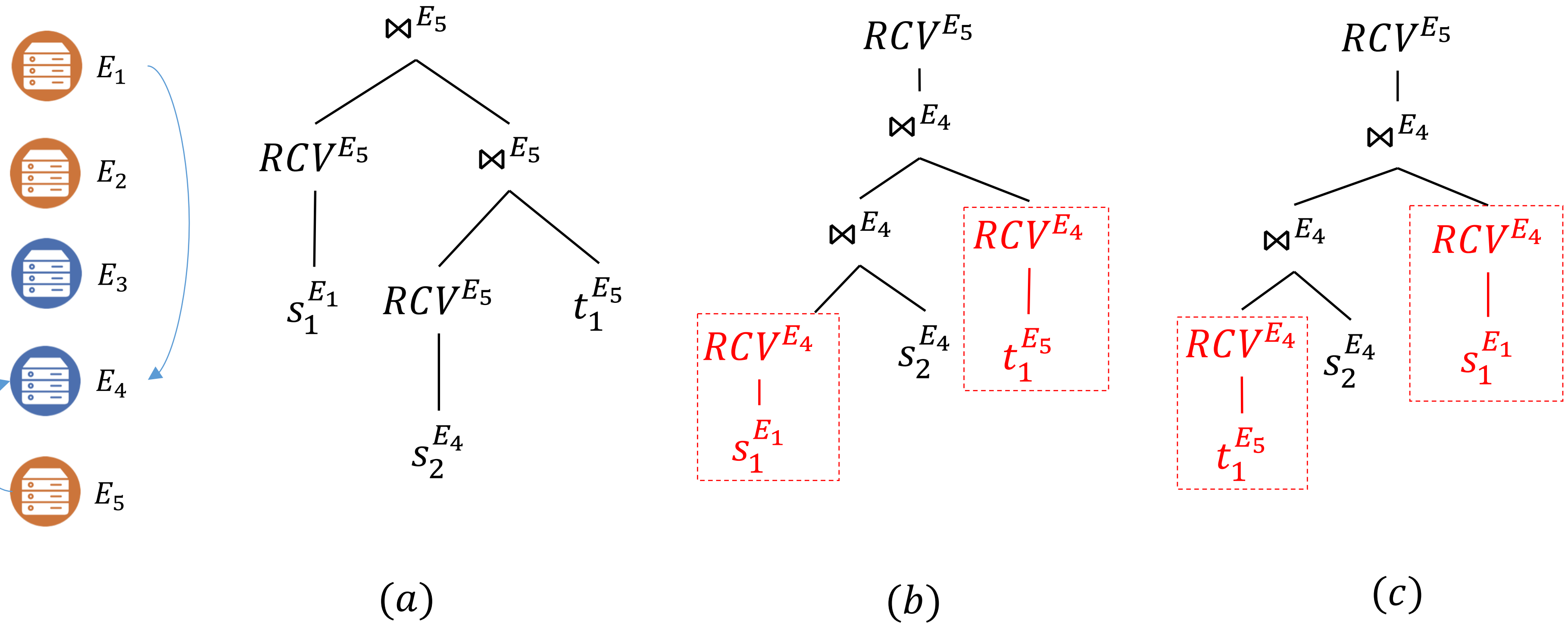
# Query Processing in Edge Computing




**SELECT** $attr_1, attr_2, attr_3, attr_4$
**FROM** $t_1$, $s_1$[RANGE 30], $s_2$[RANGE 30]
**WHERE** $t_1.s_1 = s_1.id$ **AND** $t_1.s_2 = s_2.id$

# Privacy Disclosure Risk

*Suppose that $E_4$ is controlled by the adversary who tries to collect users' private information*

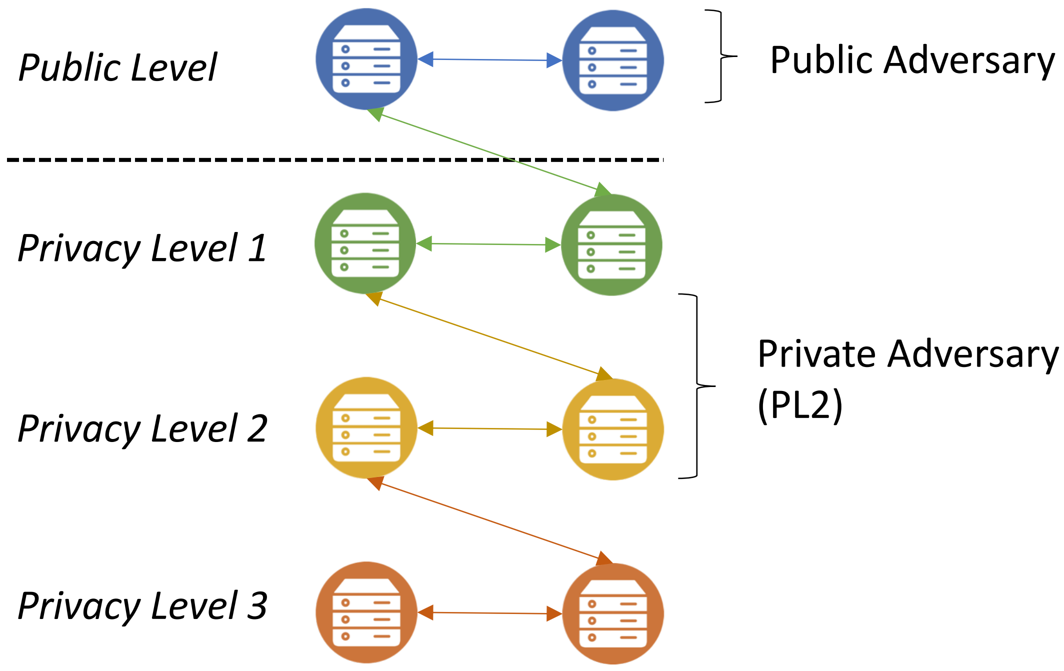



Samples of query execution plan candidates

*As a result, the adversary at the public edge node $E_4$ can acquire the intermediate sensitive data even if it does not have access to edge nodes where the sensitive data is stored.*

# Adversary Model

- Public Adversary
  - *has complete control of public edge nodes*
  - *can access any data stored in public edge nodes*
- Private Adversary
  - *can access the private edge nodes belonging to a specific privacy level*




*the adversary can access any intermediate data shipped to its controlled edge nodes during the query plan execution phase*

→ the intermediate data inference attack.

# Privacy Guarantee

- No privacy-sensitive information is disclosed in the query processing phase in the edge computing.
  - *if an adversary controls a public edge node*
    - it will not infer any privacy-sensitive information from monitoring the query operations
  - *even if the adversary controls a private edge node with privacy level p*
    - it cannot infer any sensitive information with privacy level higher than p

# Query Guard Framework

- A traditional dynamic programming enumeration skeleton
  - *the optimal plan is generated by joining optimal sub-plans in a bottom-up manner*
- Specifically
  - *Iterative dynamic programming approach*
  - *Heuristic-based methods*

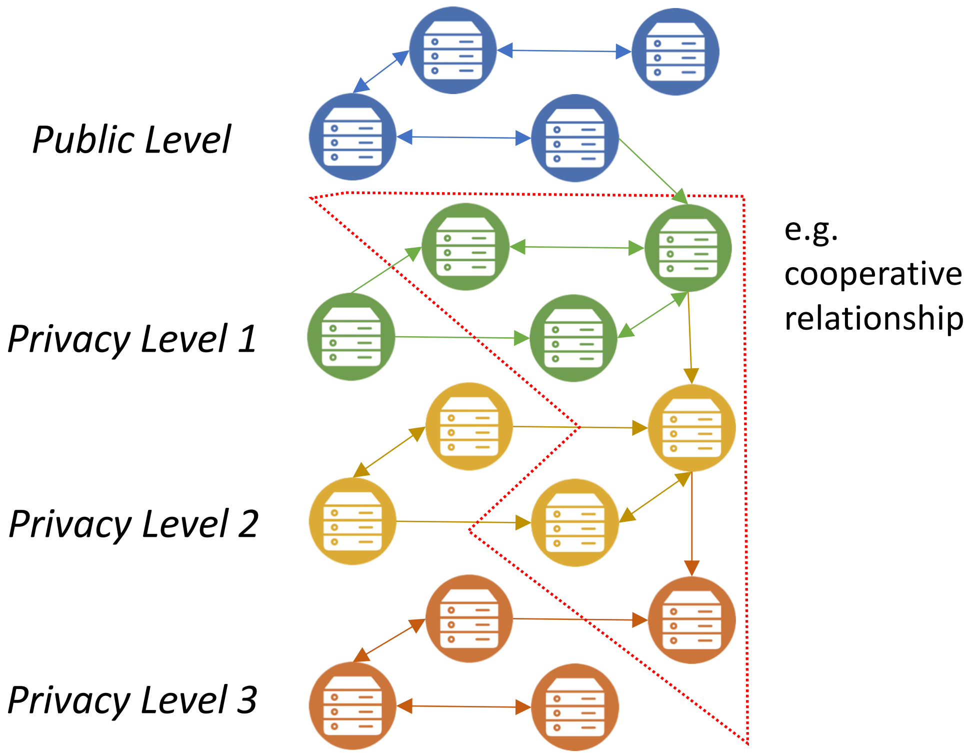
**Algorithm 1:** Pseudocode for QueryGuard framework

**Input:** A set of relations or streams $R = \{R_i\}$ with size $n$ generated from a query $Q$
**Output:** The optimal query plan

**for** $i = 1$ *to* $n$ **do**
    plans($\{R_i\}$) := access-plans($\{R_i\}$)
    LATENCY-AWARE-PRUNE(plans($\{R_i\}$))
$toDo := R$
**while** $|toDo| > 1$ **do**
    $b$ := balanced-parameter($|toDo|, k$)
    **for** $i = 2$ *to* $b$ **do**
        **forall** $S \subset R$ **and** $|S| = i$ **do**
            plans($S$) := $\emptyset$
            **forall** $O \subset S$ **and** $O \neq \emptyset$ **do**
                plans($S$) := plans($S$) $\cup$ PRIVACY-JOIN(plans($O$), plans($S \setminus O$))
                LATENCY-AWARE-PRUNE(plans($S$))
    **find** $P, V$ **with** $P \in$ plans($V$), $V \subset toDo$, $|V| = k$ **such that** eval($P$) = min{eval($P'$) $|P' \in$ plans($W$), $W \subset toDo$, $|W| = k$}
    **generate** new symbol: $\mathcal{T}$, plans($\mathcal{T}$) = $\{P\}$
    $toDo = toDo - V \cup \{\mathcal{T}\}$
    **forall** $O \subset V$ **do**
        delete(plans($O$))
finalize-plans(plans($R$))
LATENCY-AWARE-PRUNE(plans($R$))
**return** plans($R$)

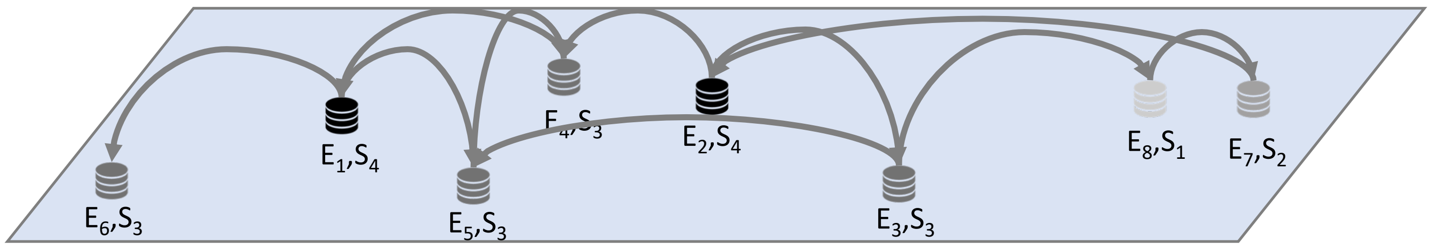
# Privacy Join

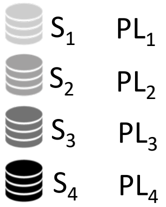
- Privacy Settings
  - *Privacy Preference*
    - the data is assigned a privacy preference parameter by data owner to control the data shipment scope
    - <u>no ship out-of-scope in join operation</u>
  - *Privacy Level*
    - each edge node is assigned a privacy level
    - the privacy level of data can be directly inferred from the privacy levels of edge nodes
    - <u>no ship down in join operation</u>



# An illustration of the critical phases in Query Guard

**Possible Joins**




*Privacy Levels: $PL_4 > PL_3 > PL_2 > PL_1$*

**Algorithm 1:** Pseudocode for QueryGuard framework

**Input:** A set of relations or streams $R = \{R_i\}$ with size $n$ generated from a query $Q$

**Output:** The optimized query plan

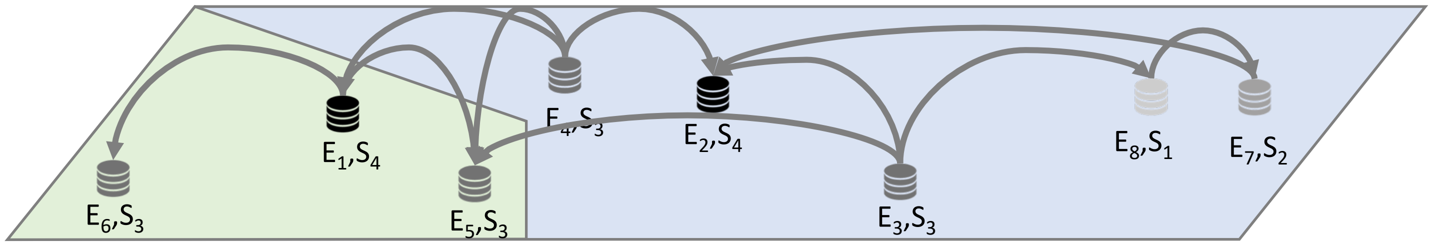
**for** $i = 1$ *to* $n$ **do**
  plans($\{R_i\}$) := access-plans($\{R_i\}$)
  LATENCY-AWARE-PRUNE(plans($\{R_i\}$))

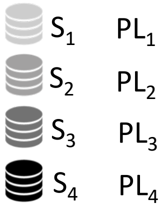
$toDo := R$
**while** $|toDo| > 1$ **do**
  $b$ := balanced-parameter($|toDo|, k$)
  **for** $i = 2$ *to* $b$ **do**
    **forall** $S \subset R$ **and** $|S| = i$ **do**
      plans($S$) := $\emptyset$
      **forall** $O \subset S$ **and** $O \neq \emptyset$ **do**
        plans($S$) := plans($S$) $\cup$ PRIVACY-JOIN(plans($O$), plans($S \setminus O$))
        LATENCY-AWARE-PRUNE(plans($S$))
  **find** $P, V$ **with** $P \in$ plans($V$), $V \subset toDo$, $|V| = k$ **such that** eval($P$) = min{eval($P'$) $|P' \in$ plans($W$), $W \subset toDo$, $|W| = k$}
  **generate** new symbol: $\mathcal{T}$, plans($\mathcal{T}$) = $\{P\}$
  $toDo = toDo - V \cup \{\mathcal{T}\}$
  **forall** $O \subset V$ **do**
    delete(plans($O$))

finalize-plans(plans($R$))
LATENCY-AWARE-PRUNE(plans($R$))
**return** plans($R$)

# An illustration of the critical phases in Query Guard

**Privacy-preserving Joins**



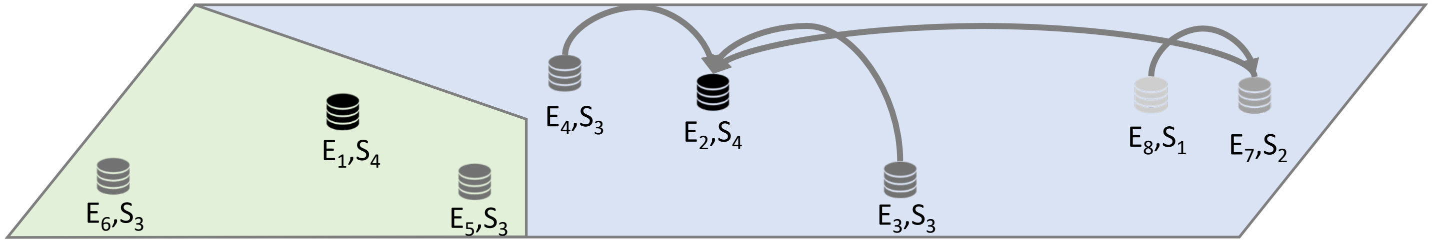

*Privacy Levels: $PL_4 > PL_3 > PL_2 > PL_1$*

**Algorithm 2:** Privacy-preserving join algorithm

```
function PRIVACY-JOIN(lplans, rplans)
    join-plans := {∅} ;
    foreach possible edge e do
        for plan l in lplans do
            if PREFERENCE-CONSTRAINT(l, e) then continue;
            lpp := LEVEL-CONSTRAINT(l, e) ;
            if lpp.flag then continue;
            for plan r in rplans do
                if PREFERENCE-CONSTRAINTE(r, e) then
                  continue;
                rpp := LEVEL-CONSTRAINT(r, e) ;
                if rpp.flag then continue;
                join := new node(lpp.root, rpp.root, e) ;
                join-plans.add(join) ;
    return join-plans
function LEVEL-CONSTRAINT(p, e)
    flag := false
    if p.root.site ≠ e then
        if p.root.P > e.P then  return (true, null) ;
        else
            rcv_node :=new node(p.root, e)
            set P of rcv_node same to P of e.
            p.root := rcv_node
    return (flag, p)
function PREFERENCE-CONSTRAINT(p, e)
    foreach  leaf node in p do
        λ := transmission threshold of leaf node
        if λ is Set type then
            if e ∉ λ then return true;
    return false
```

# An illustration of the critical phases in Query Guard

**Latency-aware Prune**




**Algorithm 3:** Latency-aware function

```
function LATENCY-AWARE-PRUNE(plans(S))
    result := {∅};
    foreach site e do t[e] := null;
    foreach plan p in plans(S) do
        c := extract the catalog information;
        if f_c(p) < f_c(t[e]) such t[e]≠null then t[e] :=p;
    foreach site e do result.add(t[e]) such t[e]≠null;
    return result
```

$S_1$ $PL_1$
$S_2$ $PL_2$
$S_3$ $PL_3$
$S_4$ $PL_4$

*Privacy Levels: $PL_4 > PL_3 > PL_2 > PL_1$*

$$f_{cost}(\mathcal{L}) = C_{cent} + \sum_{\forall (e_i,e_j)\in\mathcal{L}} (n_{bytes}^{e_i\rightarrow e_j} \cdot t_{estimate}^{e_i\rightarrow e_j})$$

$$t_{estimate}^{e_i\rightarrow e_j} = \alpha \cdot t_{avg}/n_{send}$$

$$\arctan(d_{geo}(e_i, e_j)) \cdot 2/\pi$$

# Experimental Evaluation

- General Setup
  - *Simulate a set of edge nodes with artificially injected network latency*
    - 15 edge nodes with specific geography information
    - Latency (ms) of the network traffic is estimated based on the distance (miles) using a linear model
      - y = 0.022x + 4.862

All the experiments were executed using randomly generated queries over randomly generated relations/streams that are distributed on the 15 edge nodes

EDGE NODE SIMULATION.

| Edge Node Address | Privacy Level | Geography |
|---|---|---|
| 10.0.1.{1-8} | {0,0,0,1,2,3,4,5} | Area nearby Pittsburgh, PA |
| 10.0.1.9 | 0 | Erie, PA |
| 10.0.1.10 | 1 | Philadelphia, PA |
| 10.0.1.11 | 2 | Allentown, PA |
| 10.0.1.12 | 3 | Harrisburg, PA |
| 10.0.1.13 | 0 | Cleveland, OH |
| 10.0.1.14 | 2 | Morgantown, WV |
| 10.0.1.15 | 3 | Washington D.C. |

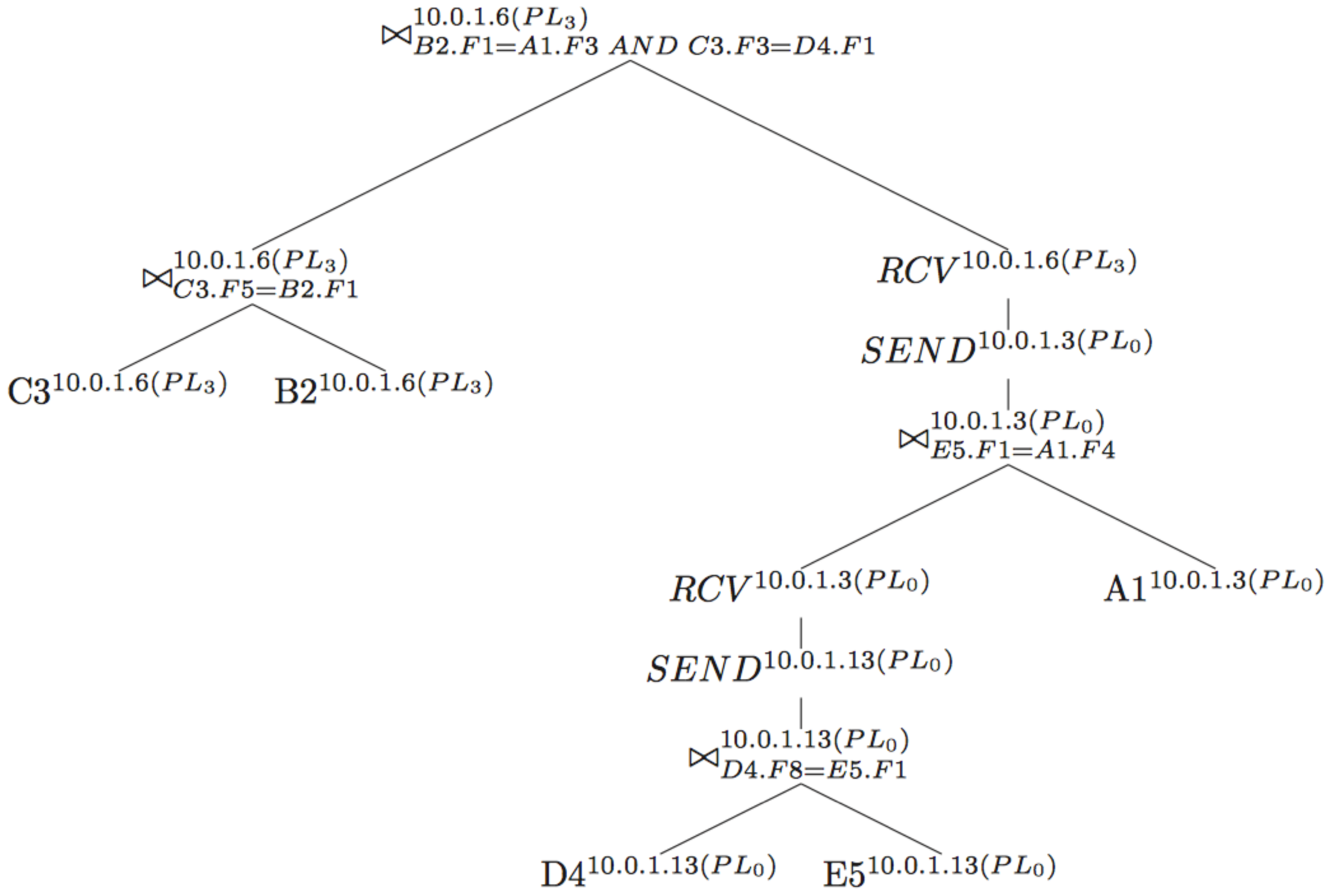
AN EXAMPLE OF RANDOMLY GENERATED RELATIONS/STREAMS.

| Relation/Stream | Edge Node | Transmission Threshold |
|---|---|---|
| A1 | 10.0.1.{3,4,5,7,10,14,15} | 10.0.1.{1-12} |
| B2 | 10.0.1.{6,8,11,12} | 10.0.1.{1-12} |
| C3 | 10.0.1.{2,6,11} | 10.0.1.{1-12} |
| D4 | 10.0.1.{2,4,5,6,11,12,13} | 10.0.1.{1-12} |
| E5 | 10.0.1.{4,12,13} | 10.0.1.{1-12} |

DISTRIBUTION OF RANDOM RELATIONS/STREAMS CARDINALITY.

| Relation Type | Cardinality of Relation | Simulation Distribution |
|---|---|---|
| I | 10-100 | 5% |
| II | 100-1000 | 15% |
| III | 1,000-10,000 | 30% |
| IV | 10,000-100,000 | 30% |
| V | 100,000-100,0000 | 15% |
| VI | 1,000,000-10,000,000 | 5% |

† The cardinality of a stream indicates the size of synopsis in DSMS.



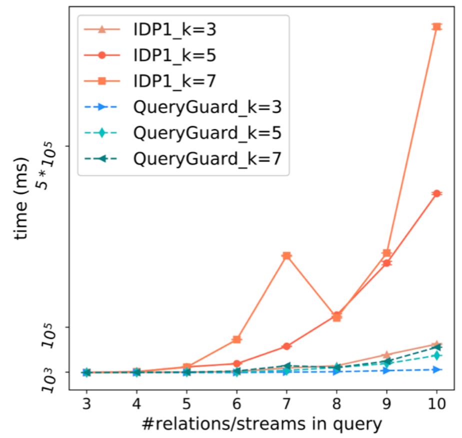

# Experimental Evaluation

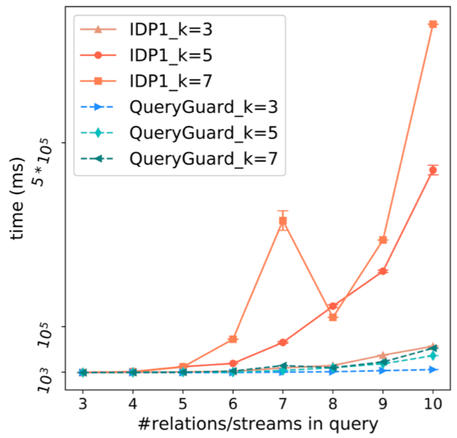
A case study of privacy-preserving processing

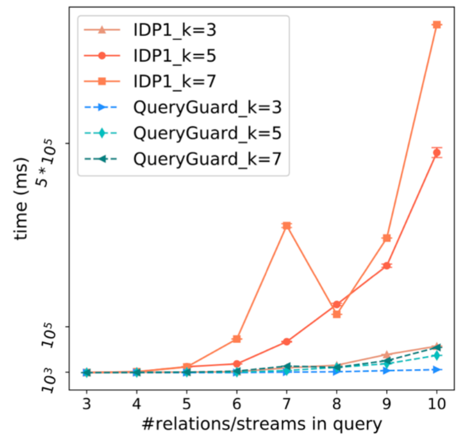
$$A1 \bowtie B2 \bowtie C3 \bowtie D4 \bowtie E5$$

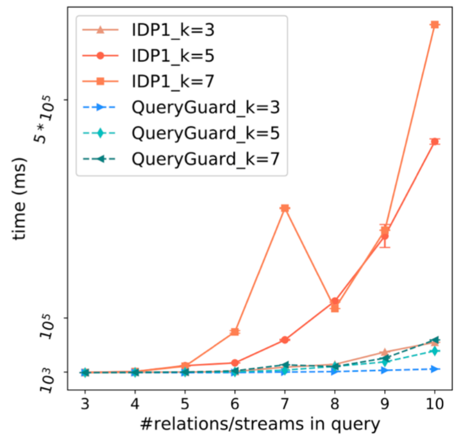
# Experimental Evaluation

- Comparison to IDP1
  - *Execution Time*
  - *our proposed technique has non-negligible performance advantage in execution time*



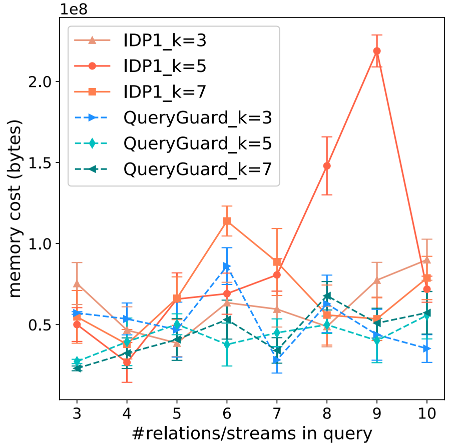



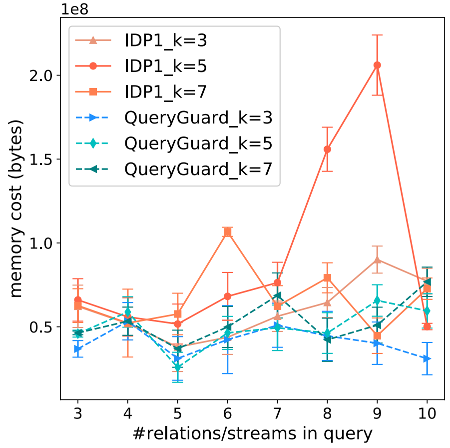



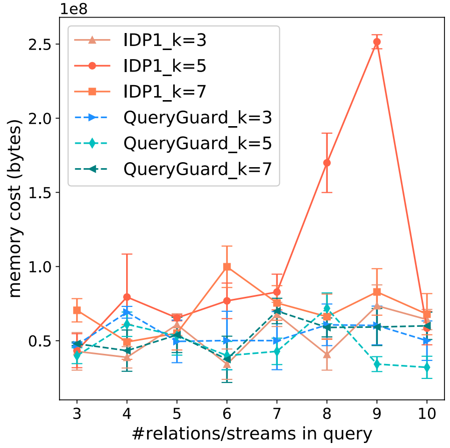



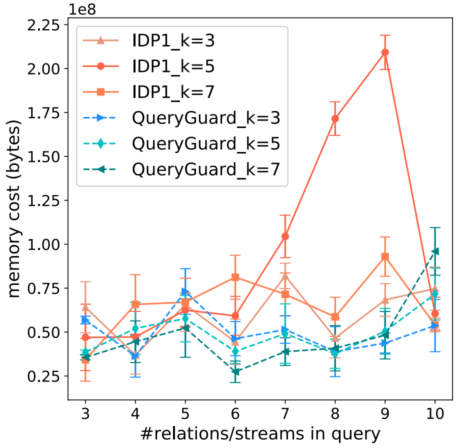
# Experimental Evaluation

- Comparison to IDP1
  - *Memory Usage*
  - *our proposed technique has non-negligible performance advantage in memory usage aspects.*



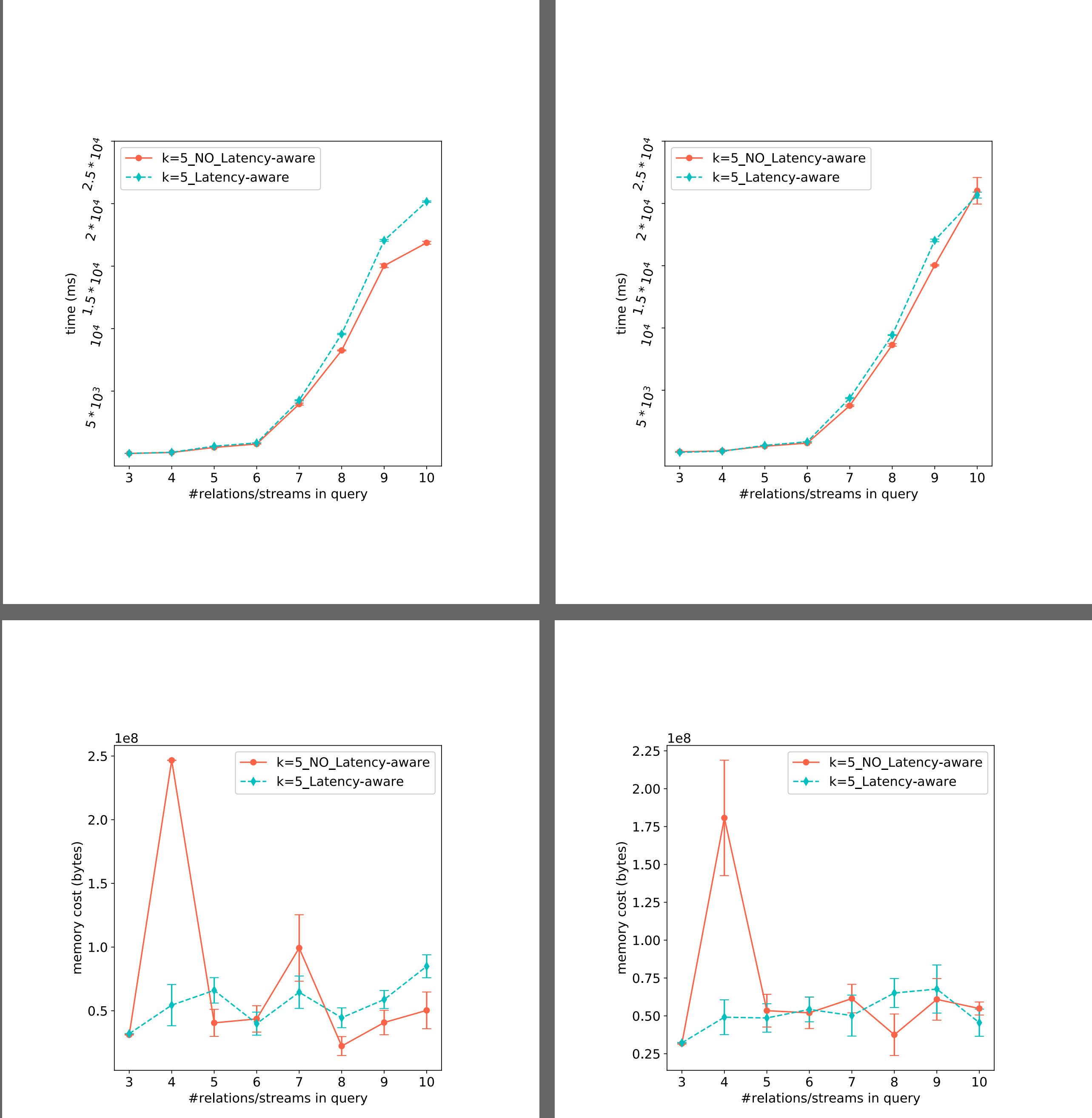



# Experimental Evaluation

- Effect of latency awareness setting
  - *to evaluate whether the latency-aware cost model influences the performance of our proposed framework*
  - *The latency-aware setting has a negligible effect on the memory usage of the algorithm, while the execution time cost has slight growth when the relation number increase.*



# Conclusion

- A privacy-preserving latency-aware query optimization framework
  - *Privacy disclosure risk analysis*
  - *Latency concerns analysis*

  - *Tackled privacy-aware and latency optimized query processing in edge computing environments*
  - *Evaluate the proposed techniques in terms of execution time and memory usage*
    - our results show that the proposed methods perform better than conventional techniques while achieving the intended privacy goals.

# Q & A

Thanks