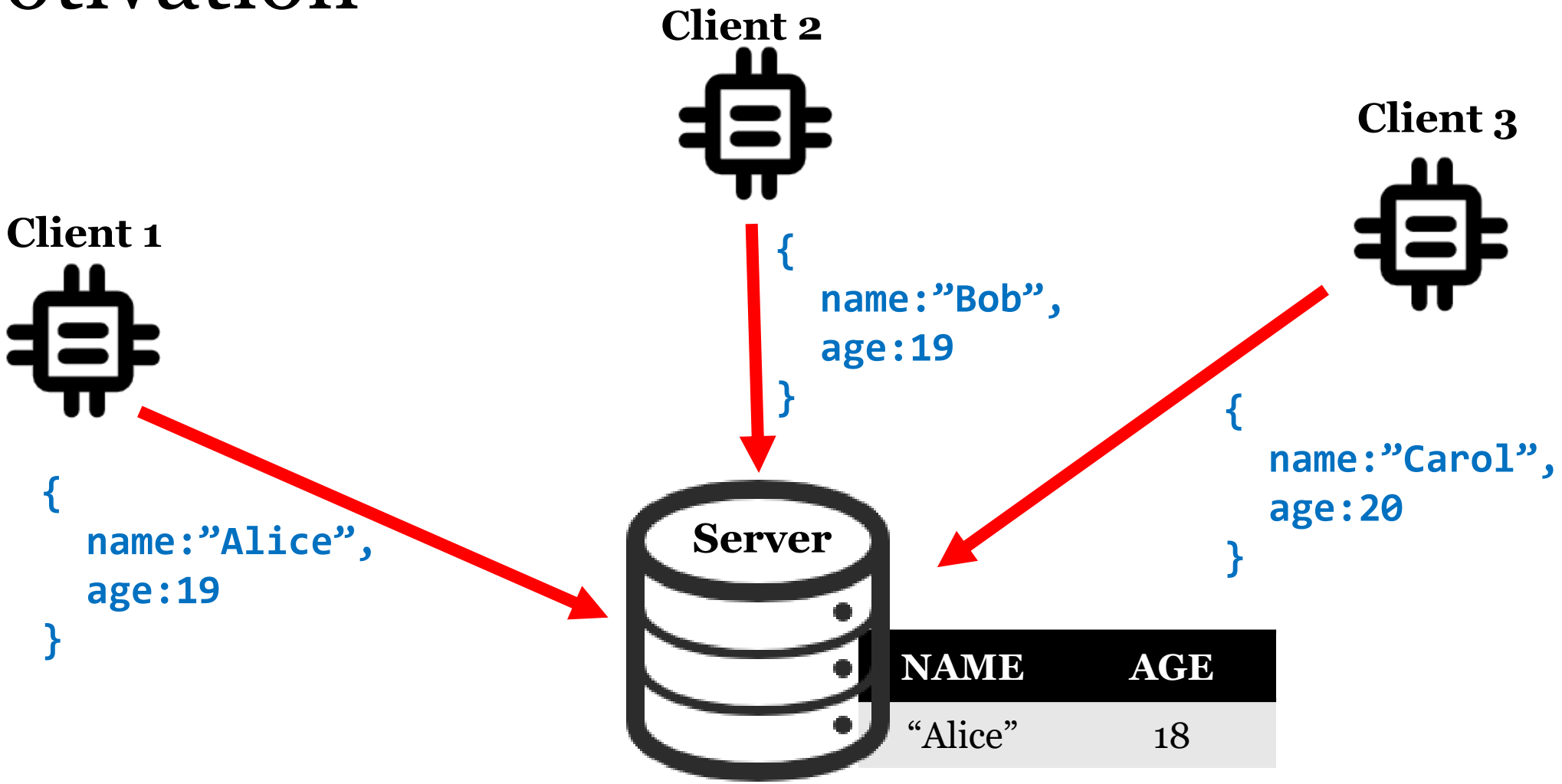
CIAO: An Optimization Framework for Client-Assisted Data Loading

Cong Ding, Dixin Tang, Xi Liang, Aaron J. Elmore, Sanjay Krishnan

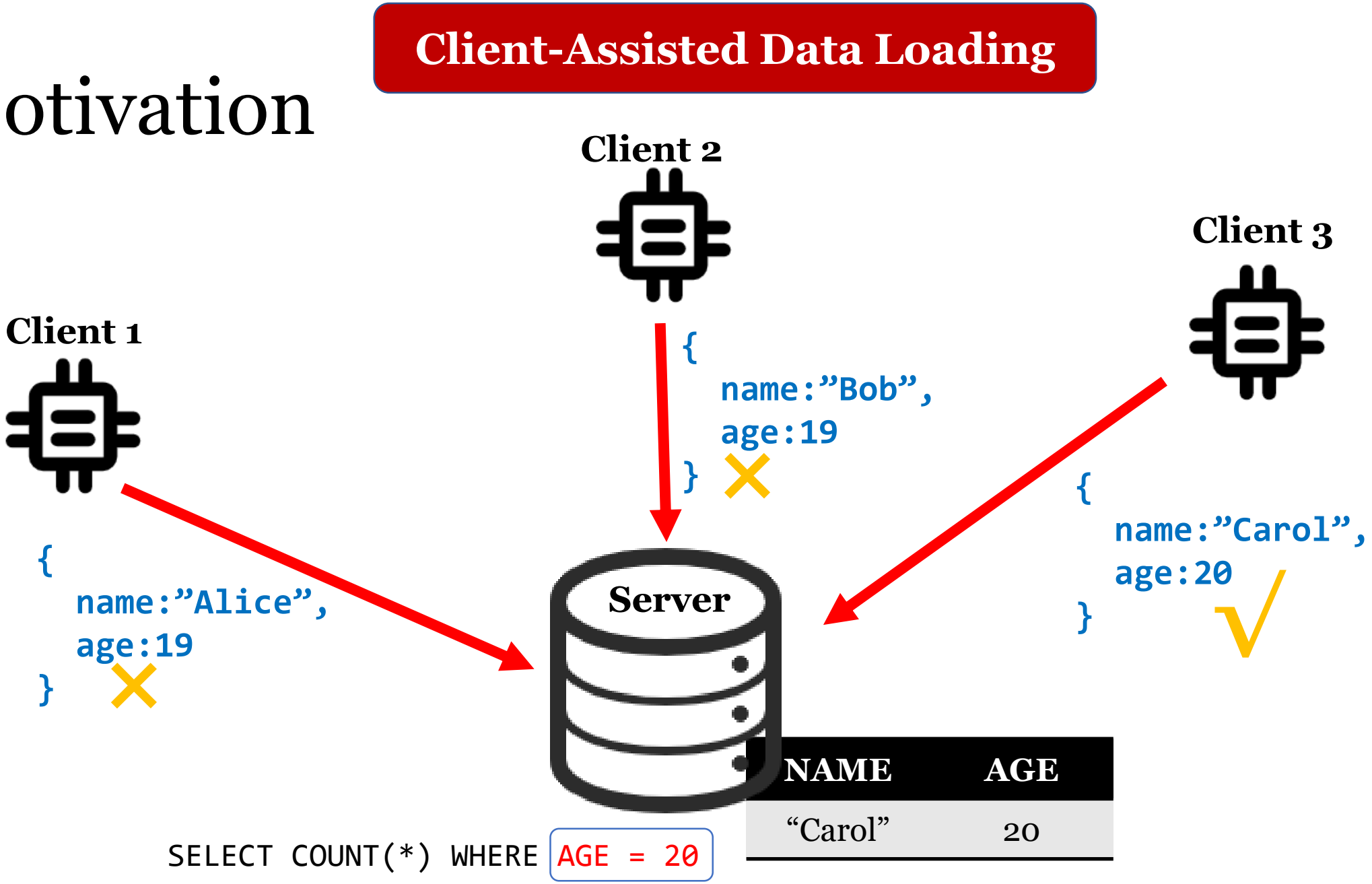
北京大学
PEKING UNIVERSITY

CHIDATA

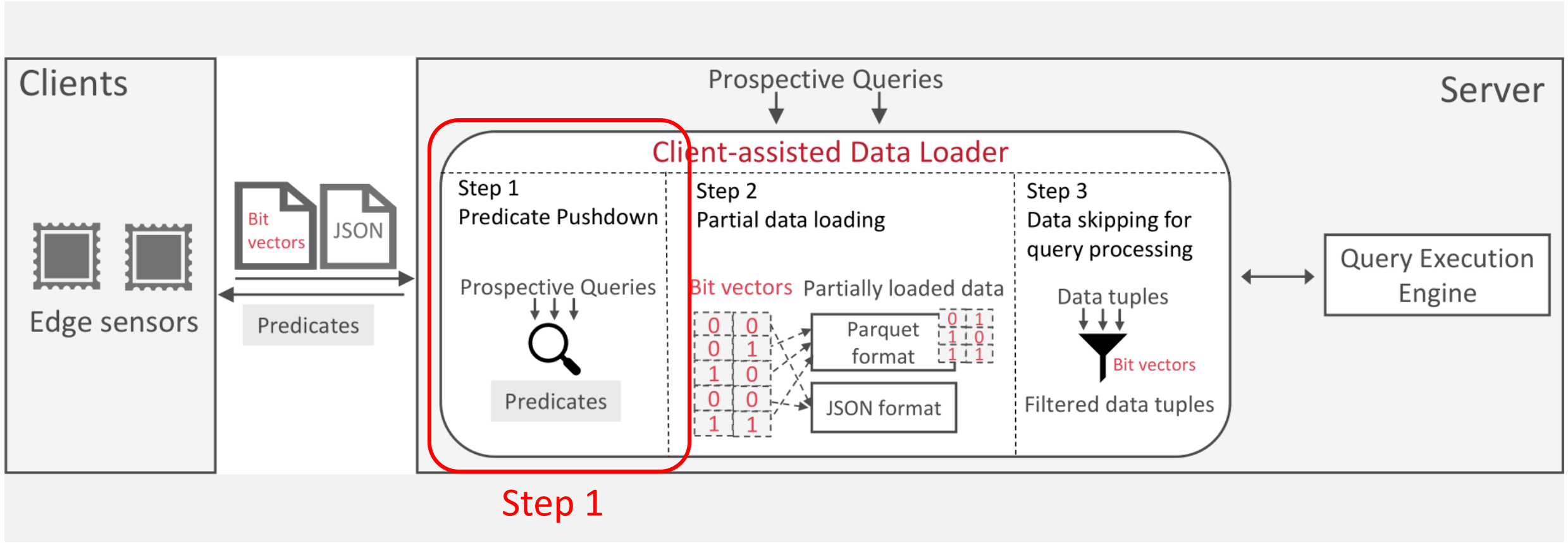
Motivation


Client 2
Client 3
Client 1
{
name:”Bob”,
age:19
}
{
name:”Carol”,
age:20
}
{
name:”Alice”,
age:19
}
Server
NAME
AGE
“Alice”
18
“Bob”
19
“Carol”
20
SELECT COUNT(*) WHERE AGE = 20

Motivation


Client-Assisted Data Loading
Client 1
Client 2
Client 3
{
name:"Alice",
age:19
}
{
name:"Bob",
age:19
}
{
name:"Carol",
age:20
}
Server
NAME
AGE
"Carol"
20
SELECT COUNT(*) WHERE AGE = 20

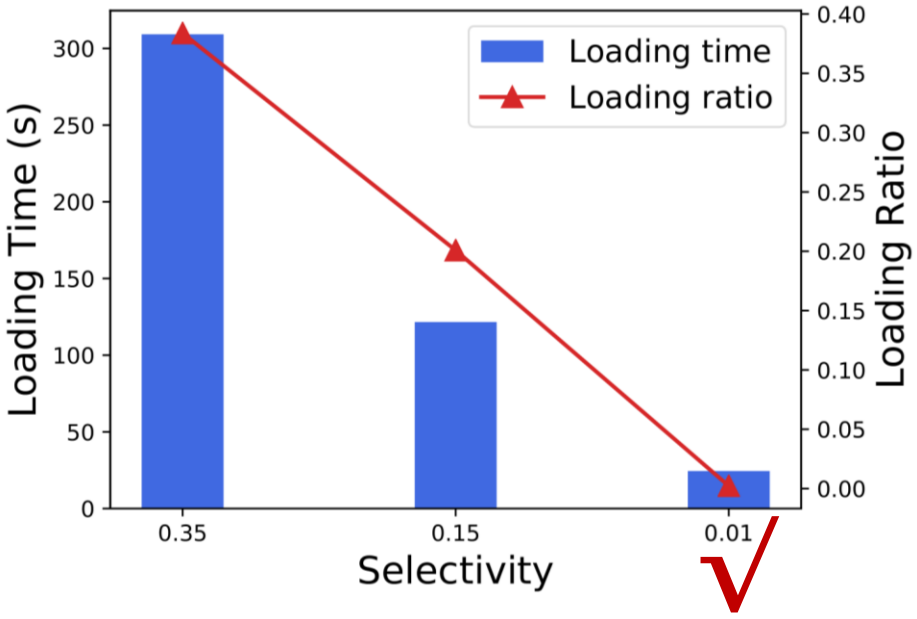
Overview


Clients
Edge sensors
Bit vectors
JSON
Predicates
Prospective Queries
Server
Client-assisted Data Loader
Step 1
Predicate Pushdown
Prospective Queries
Predicates
Step 2
Partial data loading
Bit vectors
Partially loaded data
Parquet format
JSON format
Step 3
Data skipping for query processing
Data tuples
Bit vectors
Filtered data tuples
Query Execution Engine
Step 1

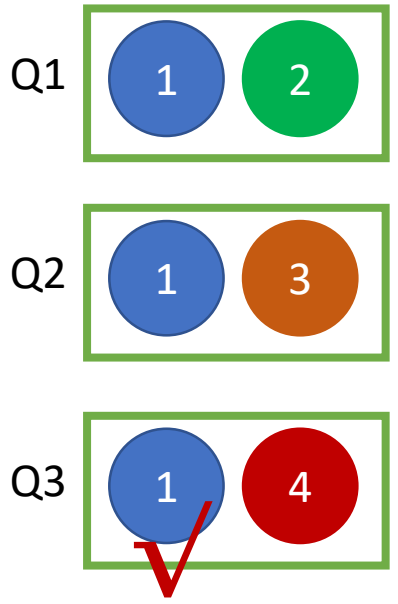
Predicate Selection

- Evaluating predicate on the clients will incur computation cost
- Choose the most *beneficial* predicate set within *limited budget*

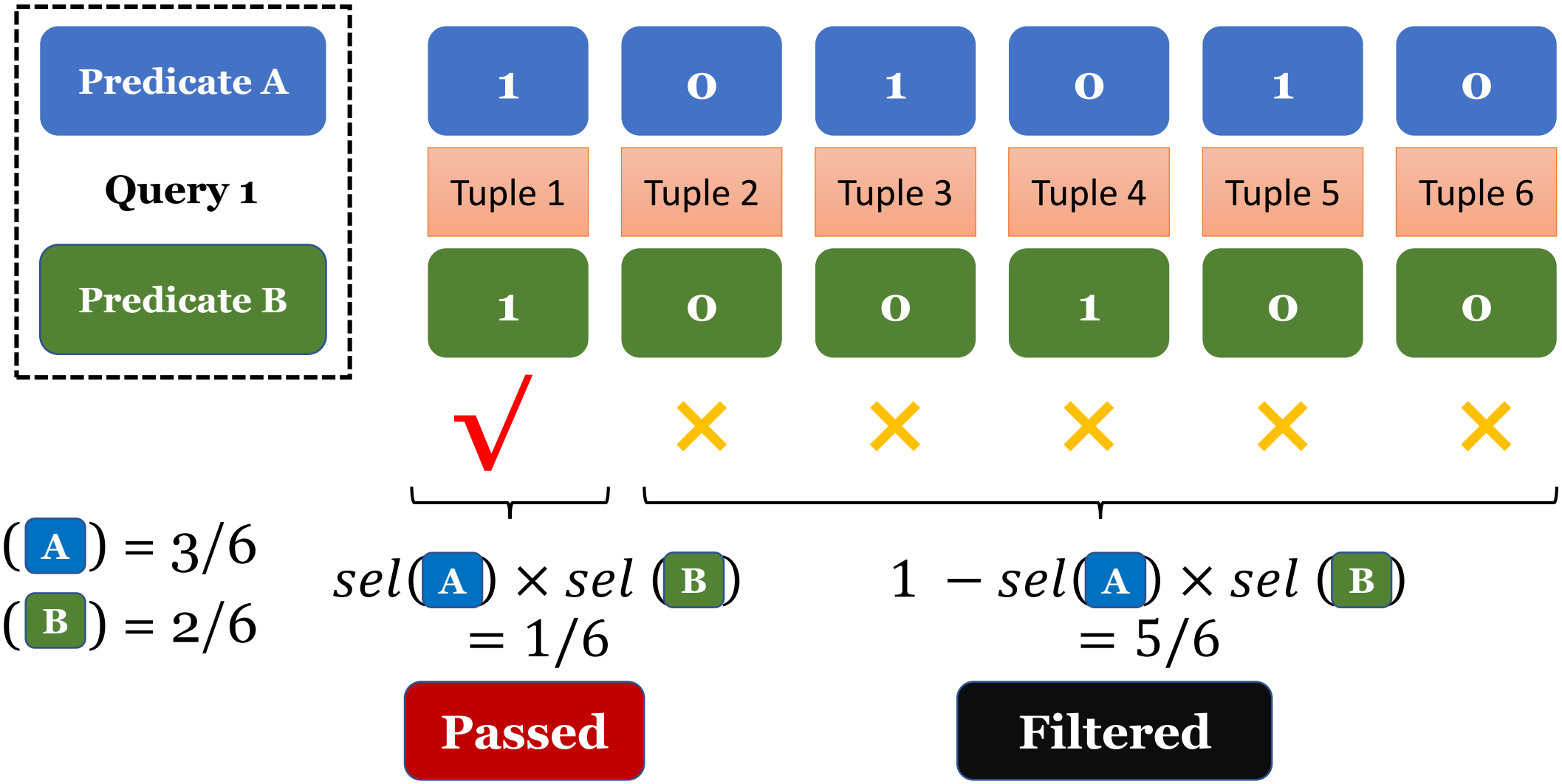
Selectivity



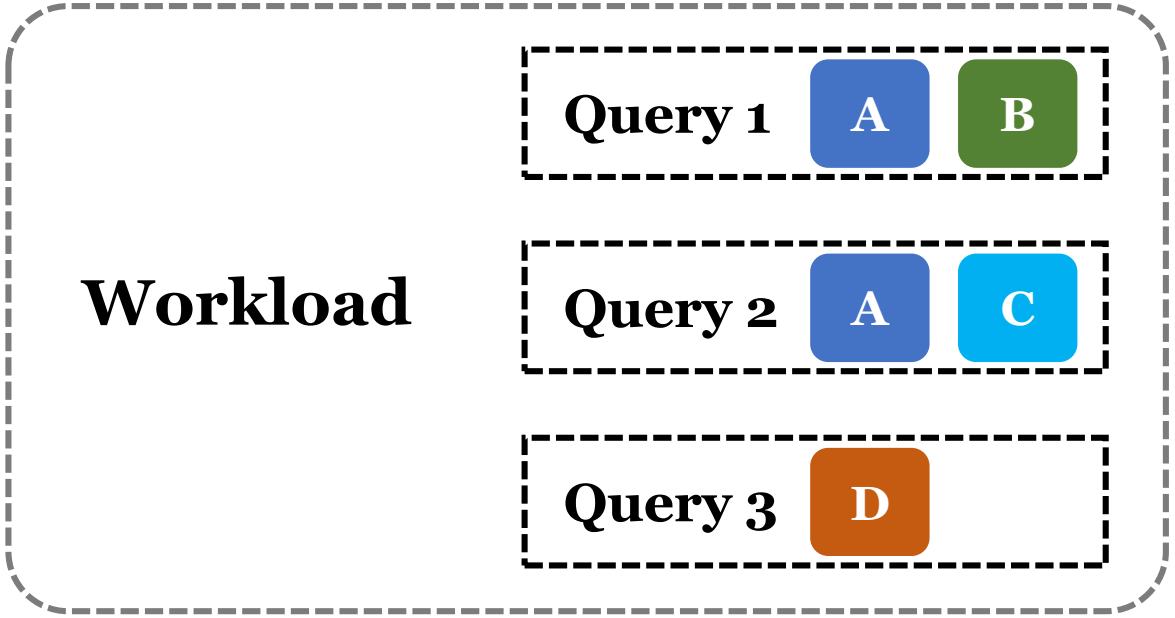
Frequency



Problem Formalization



Problem Formalization

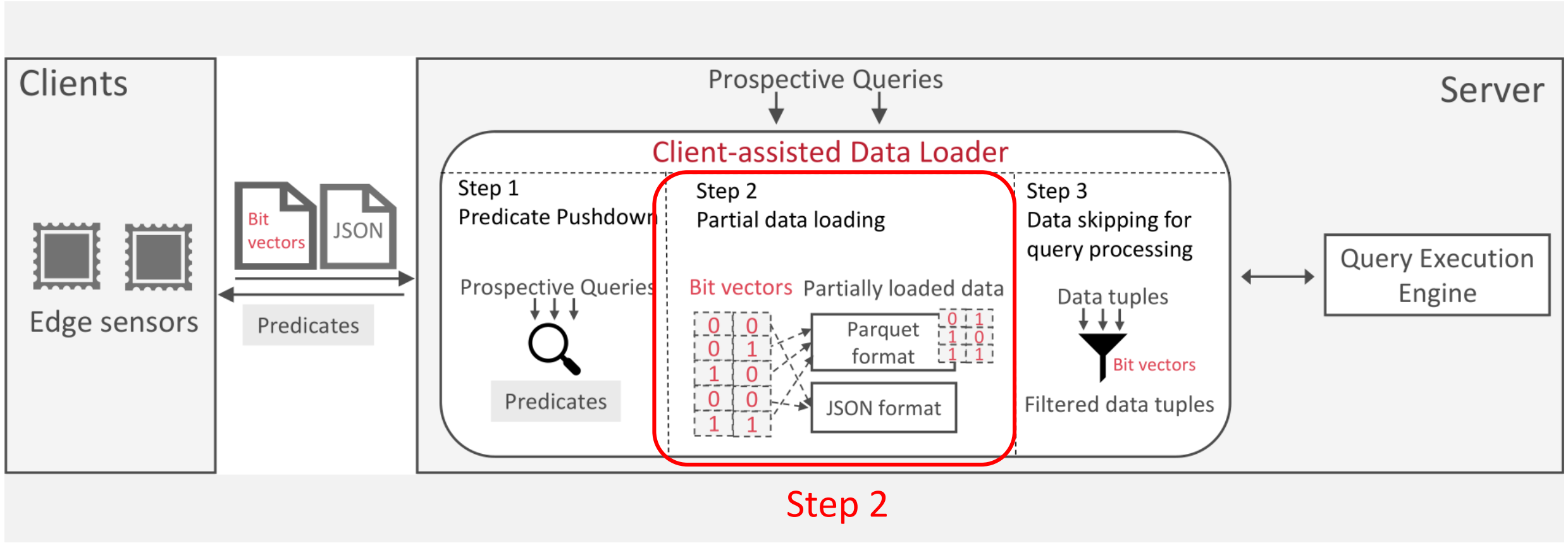


$$f = 1 - sel(A) \times sel(B)$$
$$+ 1 - sel(A) \times sel(C)$$
$$+ 1 - sel(D)$$

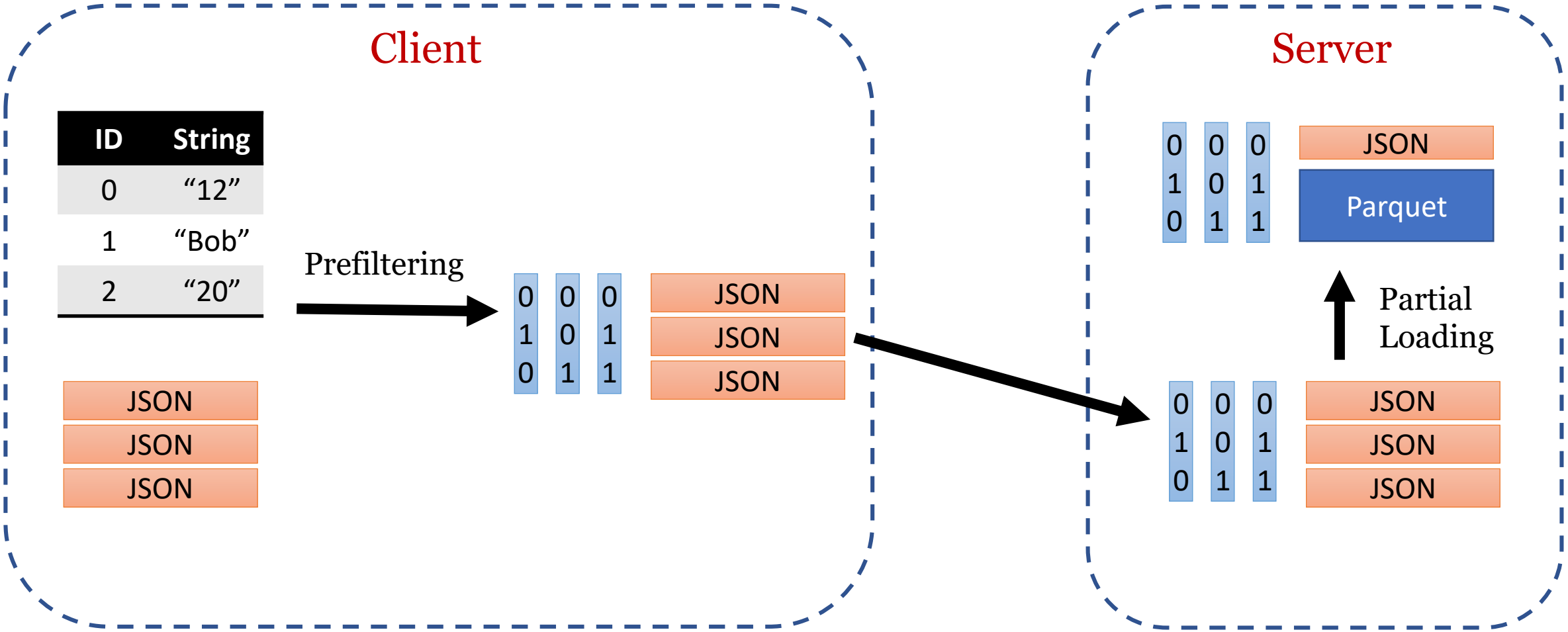
- Submodular Optimization Problem

$$\max f(S) = \sum_{q_i \in Q} [1 - \prod_{p_j \in (q_i \cap S)} sel(p_j)]$$
$$\text{s.t.} \sum_{p_i \in S} cost(p_i) \leq B$$

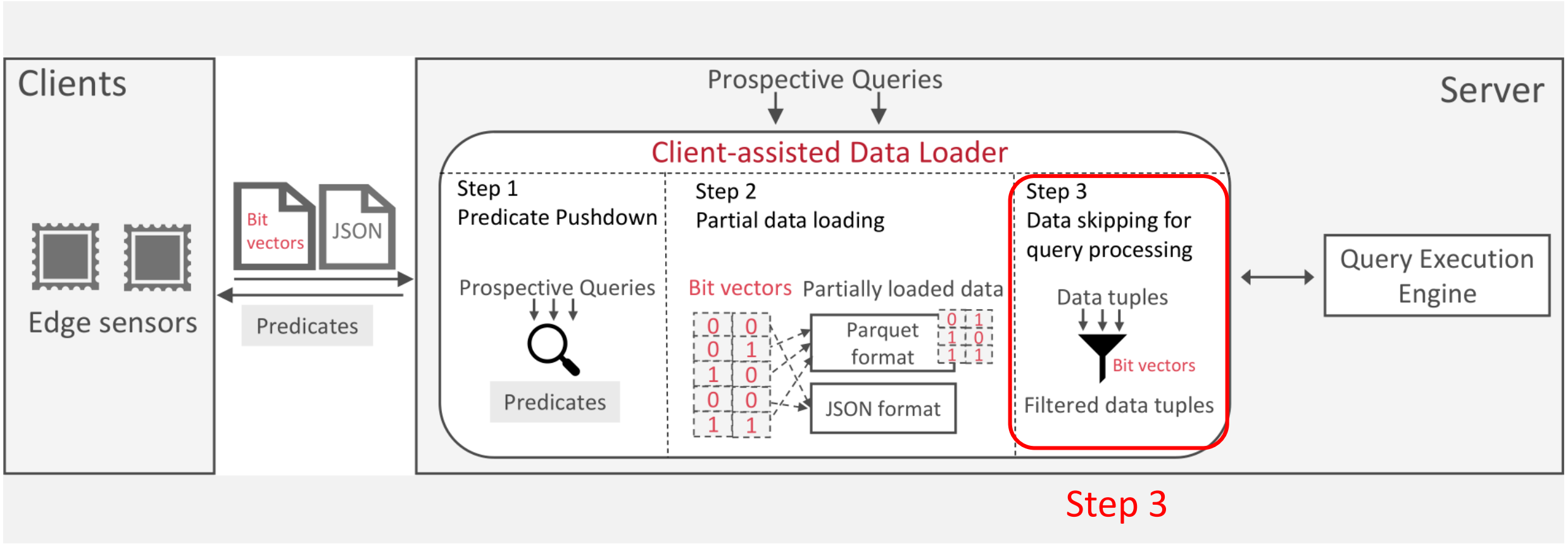
Overview


Clients
Edge sensors
Bit vectors
JSON
Predicates
Prospective Queries
Server
Client-assisted Data Loader
Step 1
Predicate Pushdown
Prospective Queries
Predicates
Step 2
Partial data loading
Bit vectors
Partially loaded data
Parquet format
JSON format
Step 3
Data skipping for query processing
Data tuples
Bit vectors
Filtered data tuples
Query Execution Engine
Step 2

Partial Data Loading

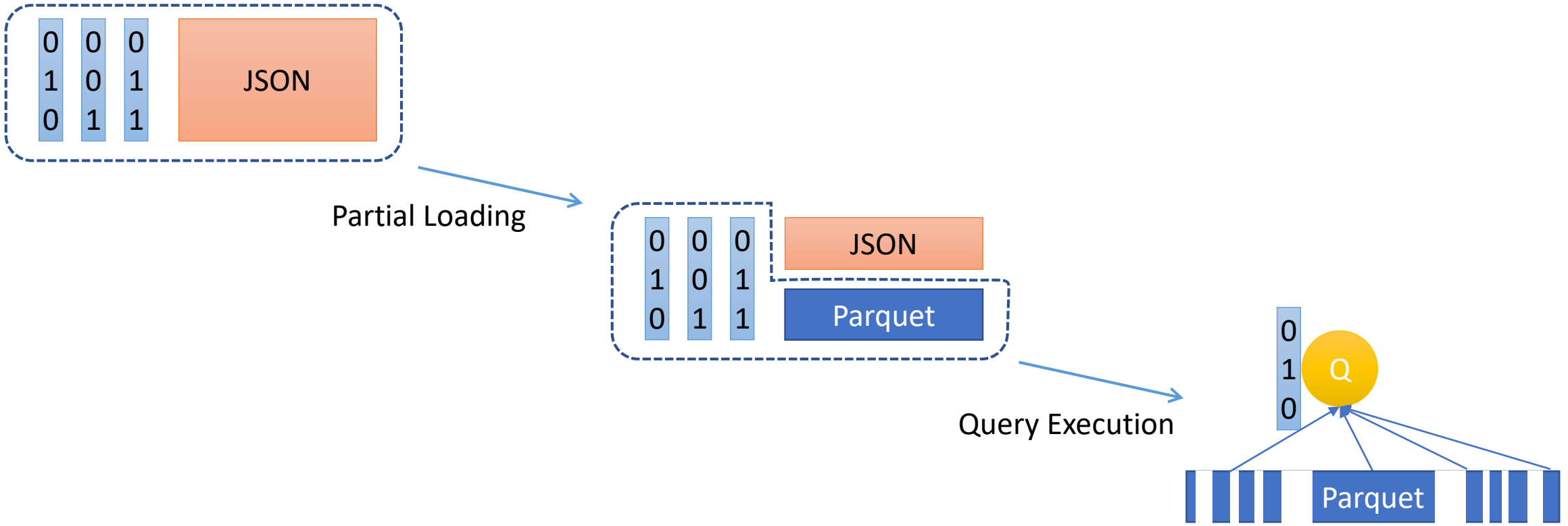


Overview


Clients
Edge sensors
Bit vectors
JSON
Predicates
Prospective Queries
Server
Client-assisted Data Loader
Step 1
Predicate Pushdown
Prospective Queries
Predicates
Step 2
Partial data loading
Bit vectors
Partially loaded data
Parquet format
JSON format
Step 3
Data skipping for query processing
Data tuples
Bit vectors
Filtered data tuples
Query Execution Engine
Step 3

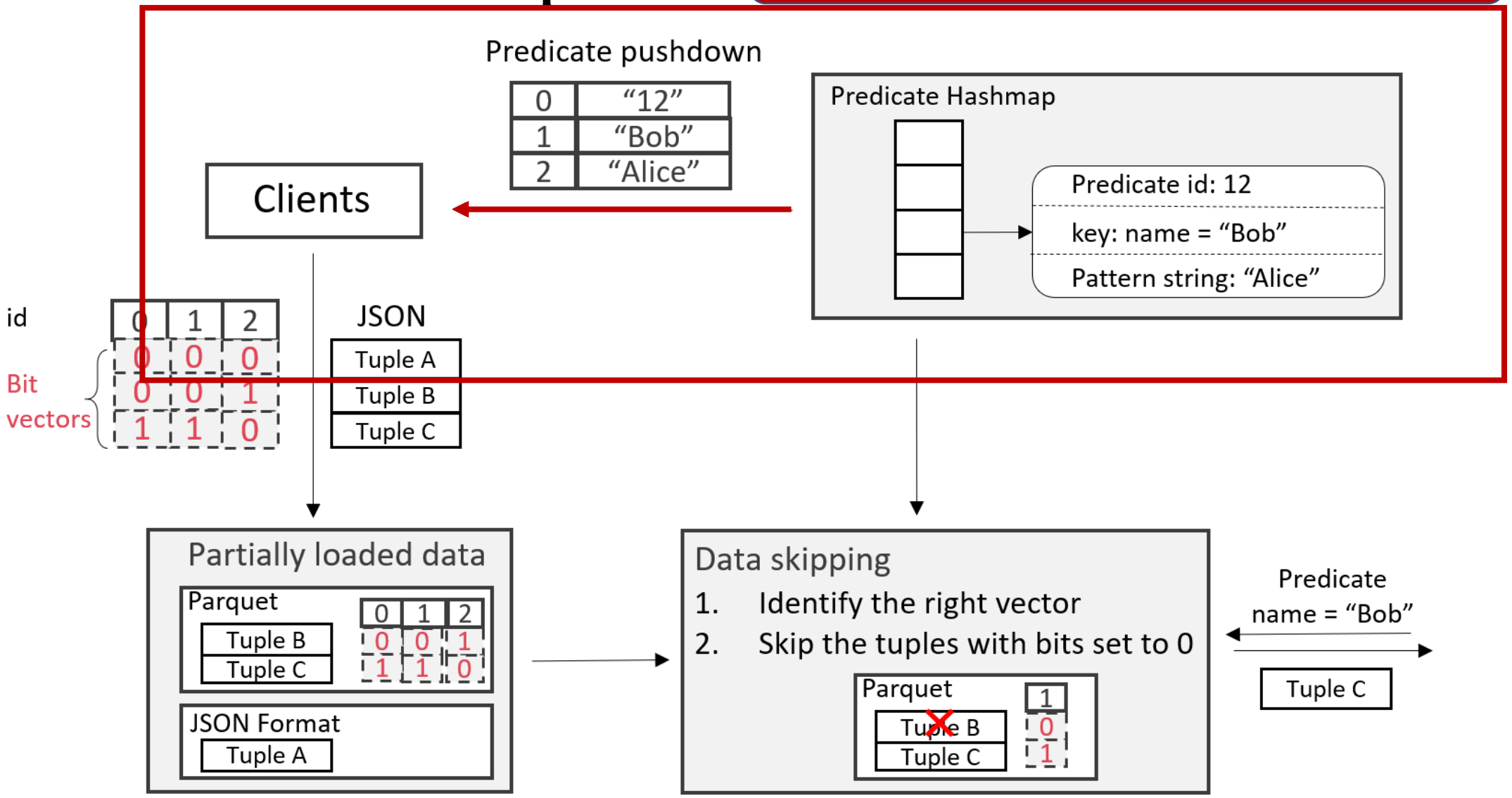
Data Skipping

- Embedded bit vectors could also help downstream query processing by skipping irrelevant tuples



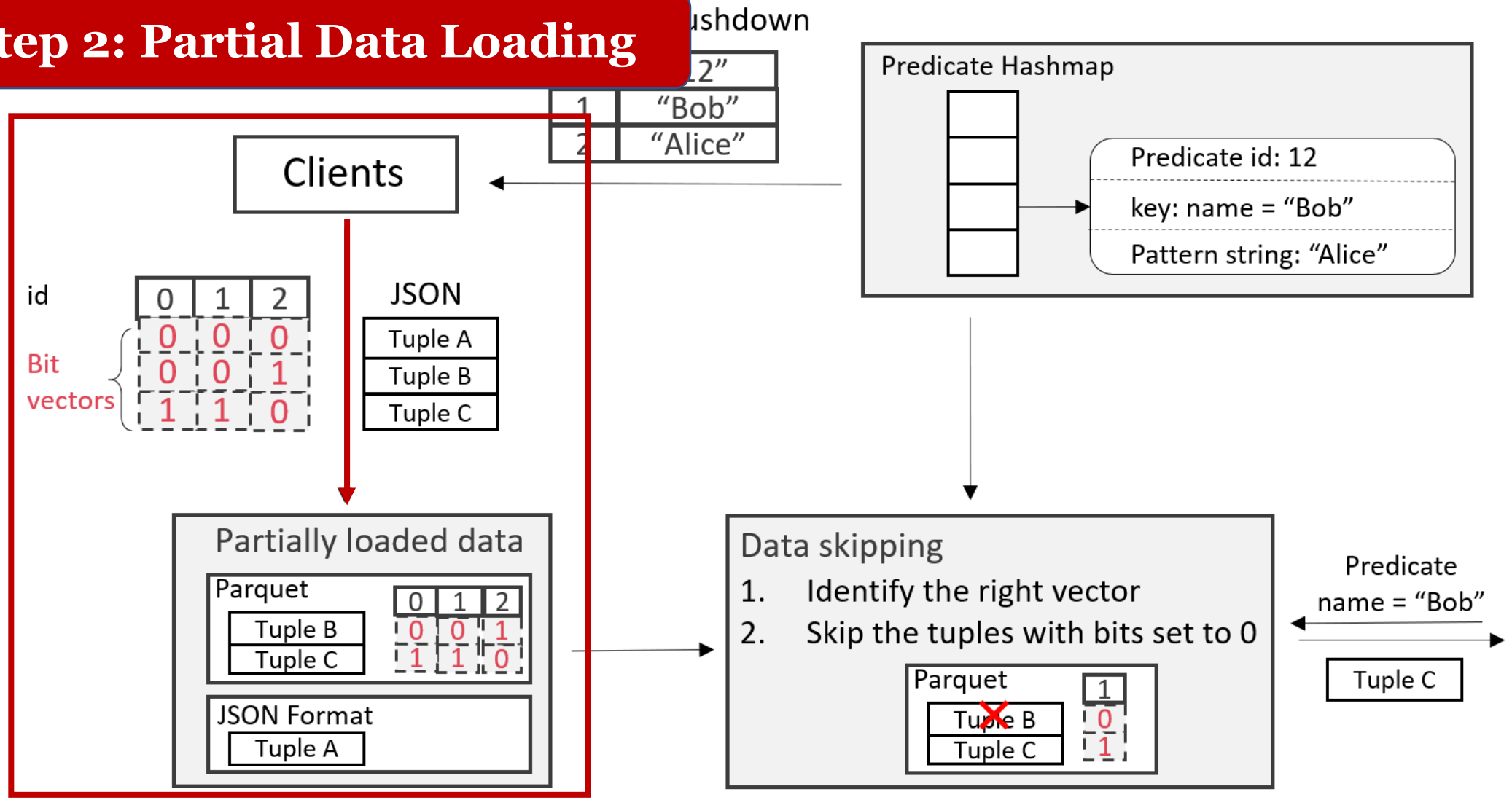
Workflow Example

Step 1: Predicate Pushdown

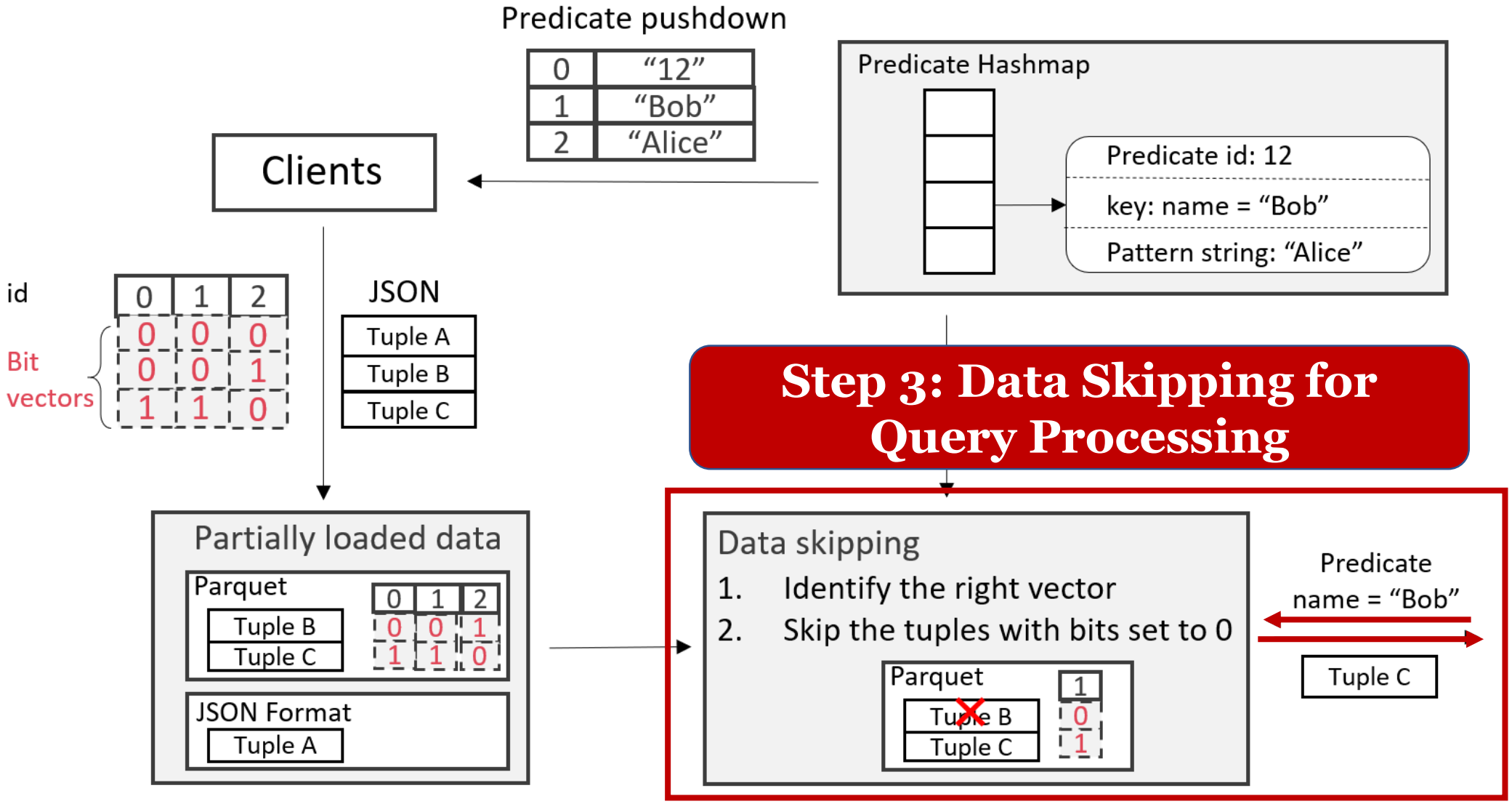


Workflow Example

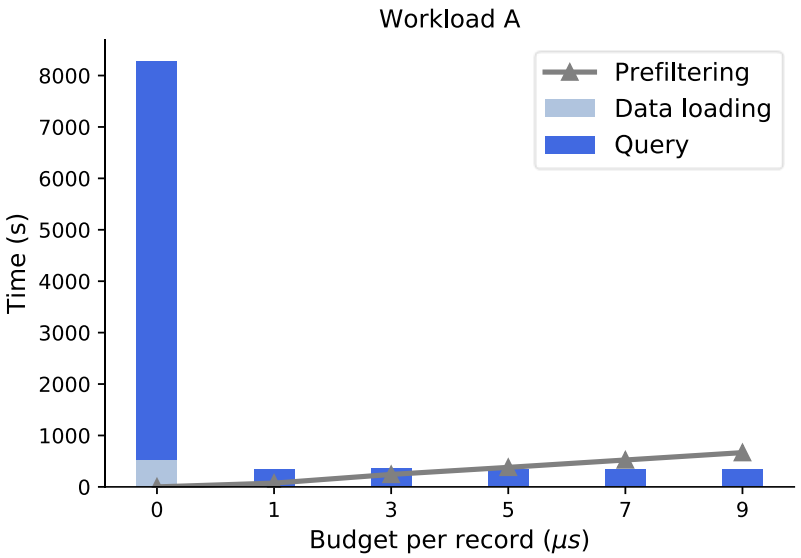
Step 2: Partial Data Loading

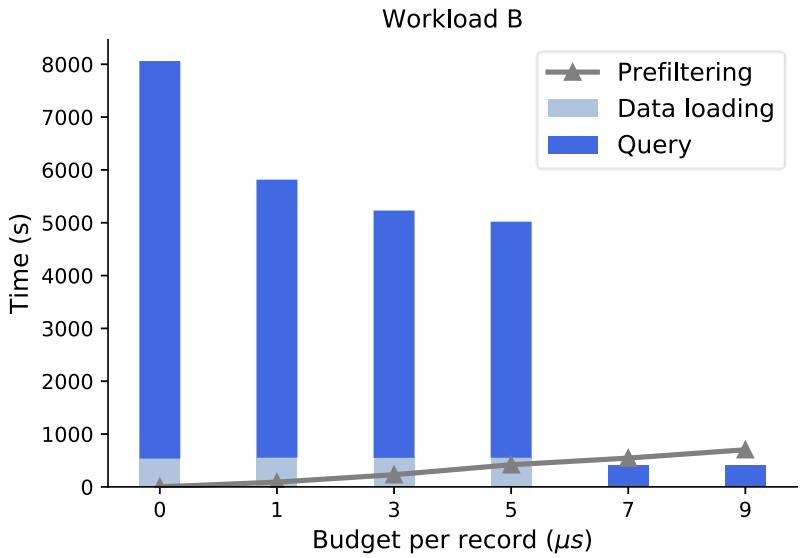


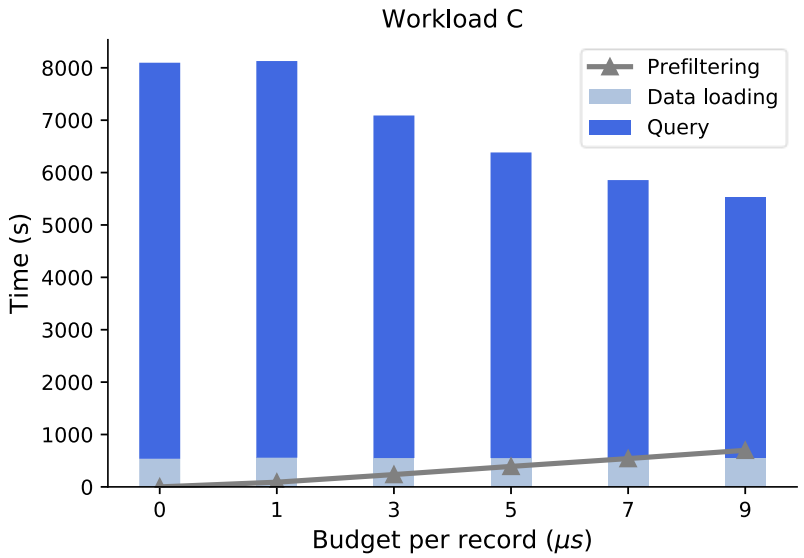
Workflow Example



End-to-End Experiments







- Dataset
  - Windows Event Log
  - 27 GB
  - ~114m Tuples

- Workloads – 200 Queries

Easy — A — B — C — Challenging

More Overlapped
More Skewed

Less Overlapped
Less Skewed

Take-Home Points from *CIAO*

- Client-Assisted Partial Loading
  - Clients will help prefilter tuples before loading.
- Near-Optimal Predicate Selection
  - Select most beneficial predicate set within limited budget.
- Data skipping
  - Downstream query processing will leverage prefiltering results.

Thanks for your attention!

Q&A