



Presented by,
MySQL AB® & O'Reilly Media, Inc.



Batched Key Access: a significant Speed-up for Join Queries

Igor Babaev
igor@mysql.com

Getting Started BKA: Introduction

Where to find the stuff (source, binaries)?

http://forge.mysql.com/wiki/Batched_Key_Access

Cloned from mysql-6.0-ndb that is currently in sync with mysql-6.0

What is a “significant speed-up” for join queries?

Depends on:

- database size
- memory available on the server
- number of concurrent connections
- **type of the join queries**

Database: DBT3 (TPC-H) : configuration **6 000 000 lineitem** records

Other tables: **part (200 000)**, **partsupp (10 000)**

Getting Started BKA: Exercise 1

```
[ENGINE=MyISAM]
```

```
<flush>
```

```
mysql> SELECT COUNT(*) FROM part, lineitem
      WHERE l_partkey=p_partkey AND p_retailprice>2050
      AND l_discount>0.04;
```

```
| 20132 |
```

```
1 row in set (4 min 15.39 sec)
```

```
mysql> SELECT COUNT(*) FROM part, lineitem WHERE ...
```

```
1 row in set (0.44 sec)
```

```
mysql> EXPLAIN SELECT COUNT(*) FROM part, lineitem WHERE l_partkey=p_partkey AND ...
```

id	select_type	table	type	ref	rows	Extra
1	SIMPLE	part	ALL	NULL	200000	Using where
1	SIMPLE	lineitem	ref	dbt3_myisam.part.p_partkey	30	Using where

Getting Started BKA: Exercise 2

```
mysql> SET join_cache_level=6;
```

<flush>

```
mysql> SELECT COUNT(*) FROM part, lineitem
        WHERE l_partkey=p_partkey AND p_retailprice>2050
        AND l_discount>0.04;
```

```
|      20132 |
```

```
1 row in set (44.41 sec)
```

```
mysql> SELECT COUNT(*) FROM part, lineitem WHERE ...
```

```
1 row in set (0.47 sec)
```

```
mysql> EXPLAIN SELECT COUNT(*) FROM part, lineitem WHERE l_partkey=p_partkey AND ...
```

id	table	type	ref	rows	Extra
1	part	ALL	NULL	200000	Using where
1	lineitem	ref	dbt3_myisam.part.p_partkey	30	Using where; Using join buffer

Getting Started BKA: Exercise 3

```
mysql> SET join_cache_level=DEFAULT;
```

```
mysql> ALTER TABLE part  
      ADD INDEX i_p_retailprice (p_retailprice);
```

```
mysql> EXPLAIN SELECT COUNT(*) FROM part, lineitem WHERE l_partkey=p_partkey AND ...
```

id	table	type	ref	rows	Extra
1	part	range	NULL	10765	Using index condition; Using MRR
1	lineitem	ref	dbt3_myisam.part.p_partkey	30	Using where

<flush>

```
mysql> SELECT COUNT(*) FROM part, lineitem  
      WHERE l_partkey=p_partkey AND p_retailprice>2050  
      AND l_discount>0.04;
```

1 row in set (4 min 12.13 sec)

Getting Started BKA: Exercise 4

```
mysql> SET optimizer_use_mrr=0;
```

```
mysql> EXPLAIN SELECT COUNT(*) FROM part, lineitem WHERE l_partkey=p_partkey AND ...
```

id	table	type	ref	rows	Extra
1	part	range	NULL	10765	Using index condition
1	lineitem	ref	dbt3_myisam.part.p_partkey	30	Using where

<flush>

```
mysql> SELECT COUNT(*) FROM part, lineitem
      WHERE l_partkey=p_partkey AND p_retailprice>2050
      AND l_discount>0.04;
```

```
| 20132 |
```

```
1 row in set (4 min 15.44 sec)
```

```
mysql> SET optimizer_use_mrr=1;
```

Getting Started BKA: Exercise 5

```
mysql> SET join_cache_level=6;
```

```
mysql> EXPLAIN SELECT COUNT(*) FROM part, lineitem WHERE l_partkey=p_partkey AND ...
```

id	table	type	ref	rows	Extra
1	part	range	NULL	10765	Using index condition; Using MRR
1	lineitem	ref	dbt3_myisam.part.p_partkey	30	Using where; Using join buffer

<flush>

```
mysql> SELECT COUNT(*) FROM part, lineitem
      WHERE l_partkey=p_partkey AND p_retailprice>2050
      AND l_discount>0.04;
```

```
| 20132 |
1 row in set (45.17 sec)
```

Getting Started BKA: Exercise 6

```
mysql> SET join_buffer_size=1024*256;
```

<flush>

```
mysql> SELECT COUNT(*) FROM part, lineitem
        WHERE l_partkey=p_partkey AND p_retailprice>2050
        AND l_discount>0.04;
```

```
|      20132 |
1 row in set (32.01 sec)
```

```
mysql> SET join_buffer_size=1024*512;
```

<flush>

```
mysql> SELECT COUNT(*) FROM part, lineitem ...
```

```
|      20132 |
1 row in set (16.72 sec)
```

Getting Started BKA: Exercise 7

```
mysql> ALTER TABLE part DROP INDEX i_p_retailprice;
```

```
mysql> SET join_buffer_size=DEFAULT;
```

```
mysql> SELECT @@join_buffer_size;
```

<flush>

```
mysql> SELECT COUNT(*) FROM part, lineitem
        WHERE l_partkey=p_partkey AND p_retailprice>2050
        AND l_discount>0.04;
```

1 row in set (46.70 sec)

```
mysql> SET join_buffer_size=1024*512;
```

<flush>

```
mysql> SELECT COUNT(*) FROM part, lineitem ...
```

```
| 20132 |
```

1 row in set (15.15 sec)

Getting Started BKA: Exercise 8

```
[ENGINE=InnoDB]
```

```
<flush>
```

```
mysql> SELECT COUNT(*) FROM part, lineitem ...
```

```
| 20132 |
```

```
1 row in set (43.23 sec)
```

```
mysql> SET join_cache_level=DEFAULT;
```

```
<flush>
```

```
mysql> SELECT COUNT(*) FROM part, lineitem ...
```

```
| 20132 |
```

```
1 row in set (3 min 56.46 sec)
```

Presented by



O'REILLY

Getting Started BKA: Conclusions

With NL Joins fetching a lot of data from big tables is slow, because **random** access of data is inevitable.

A possible solution would be to create additional indexes to cover the fetched data columns. Yet, in many situation it's not acceptable.

Batched Key Access Join executes with no random accesses to fetch data. That's why it's much faster with “cold” tables or with huge tables for which only a fraction of data is expected to be in cache.

The larger join buffer is used for BKA, the better is the performance. A significant speed-up can be achieved with the join buffer of quite a reasonable size (**1-2 MB**).

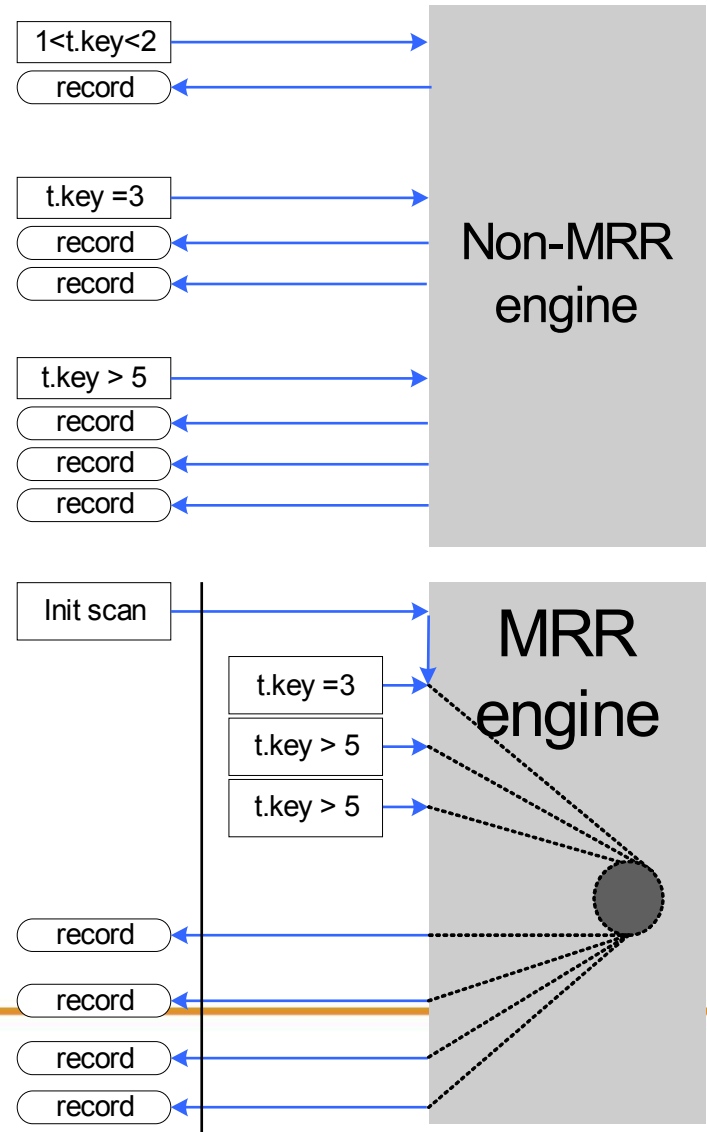
The Idea of Multi-Range Read

Non-MRR:

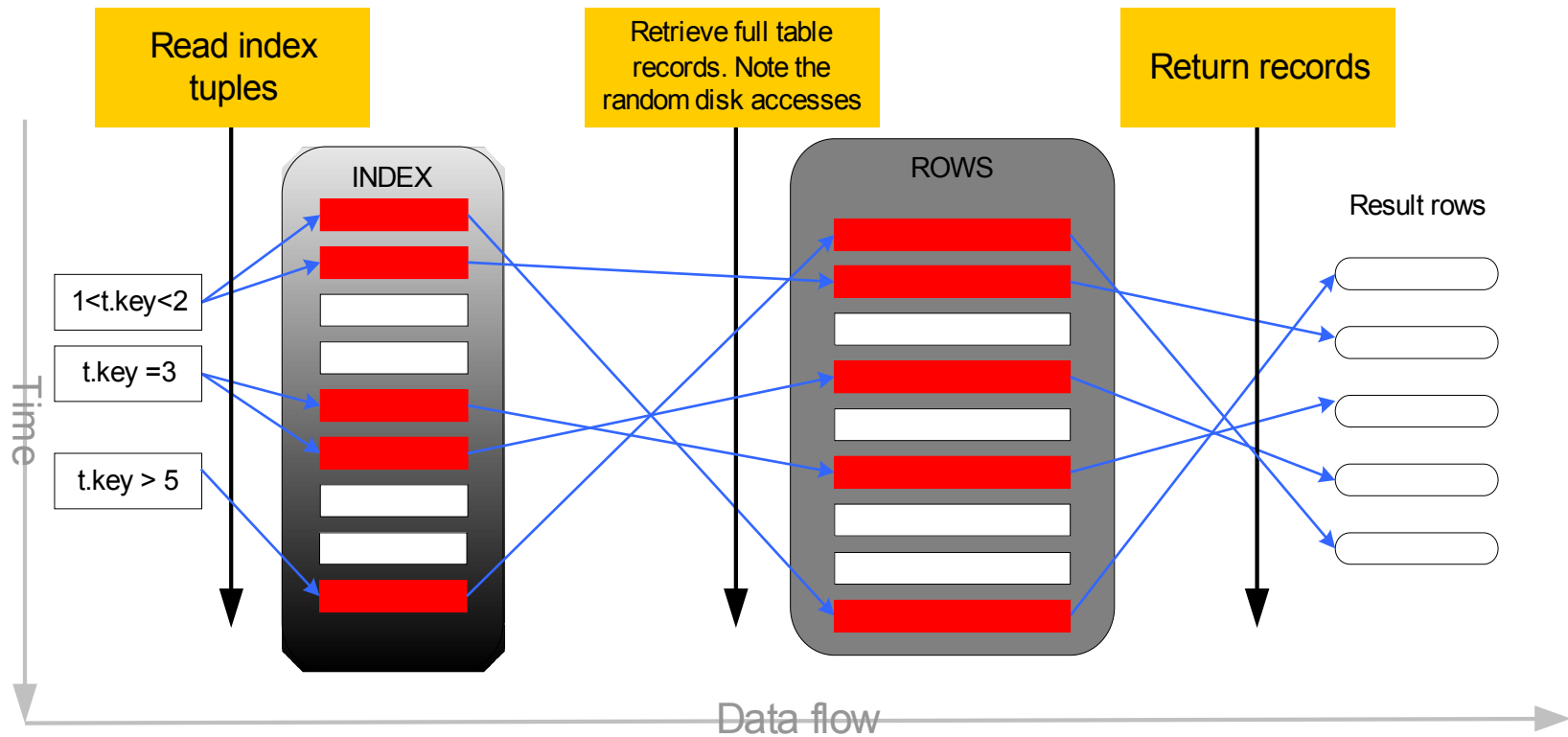
- At least one roundtrip per scanned range.
- Engine is forced to access index entries/table data in pre-determined order.

MRR:

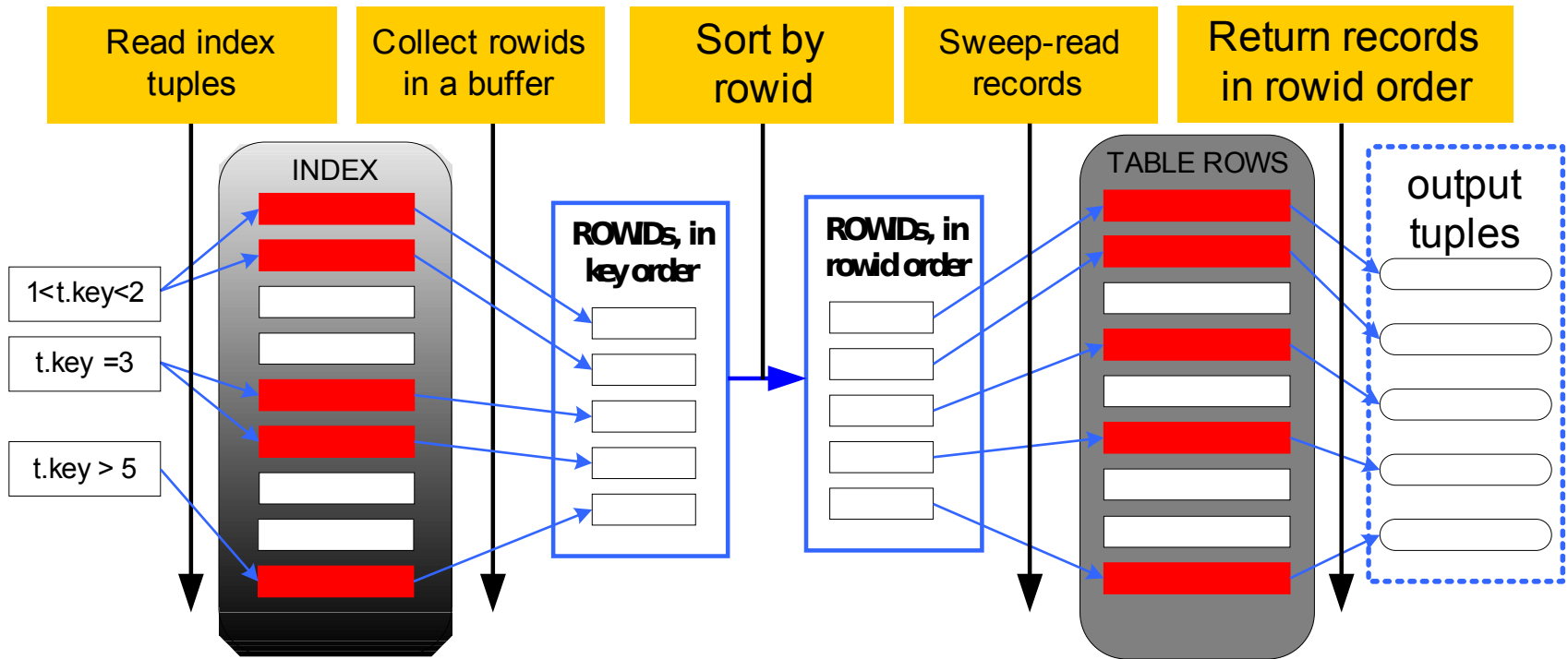
- The number of roundtrips can be reduced to one.
- Engine can fetch data rows in an optimized order.



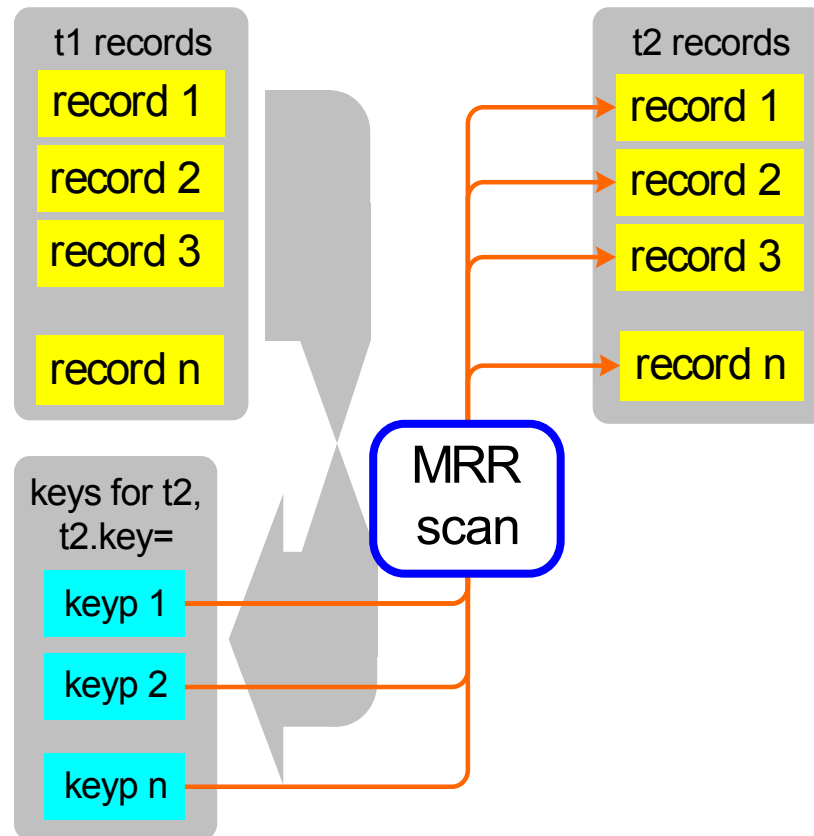
Data access without MRR (MyISAM)



Data Access with MRR (MyISAM)

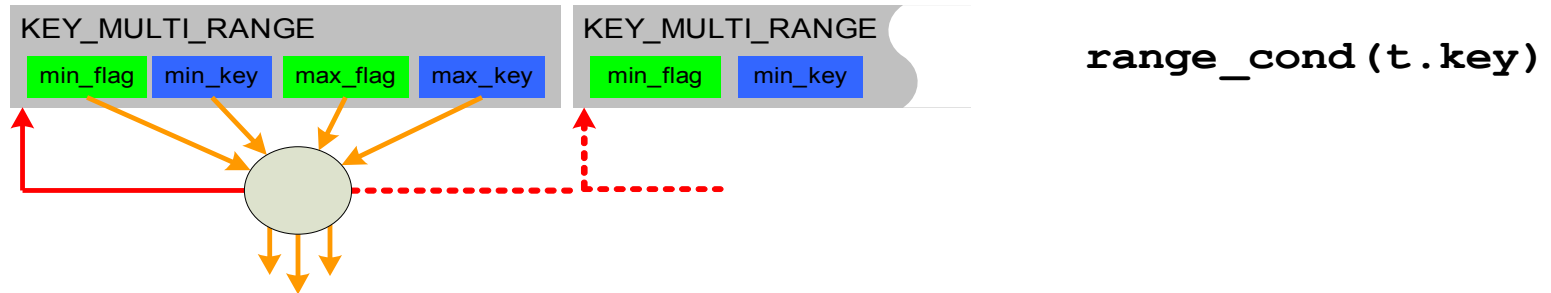


The Idea Of Batched Key Access Join

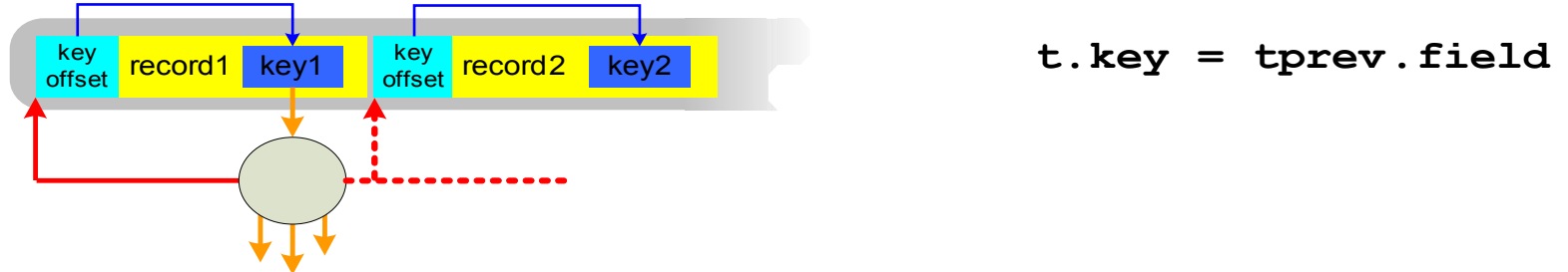


Batched Key Access: Join Buffer

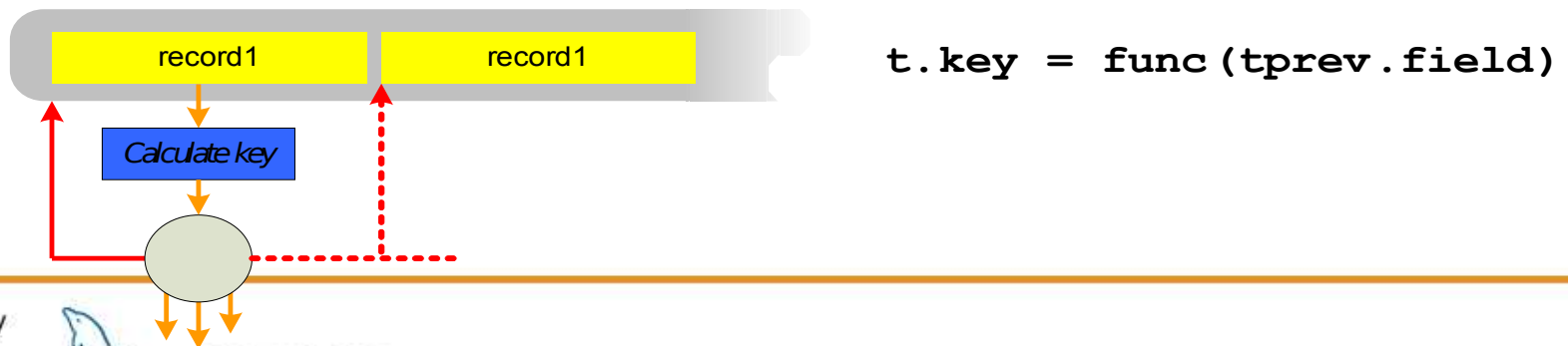
- “range”: array of “generic” interval structures



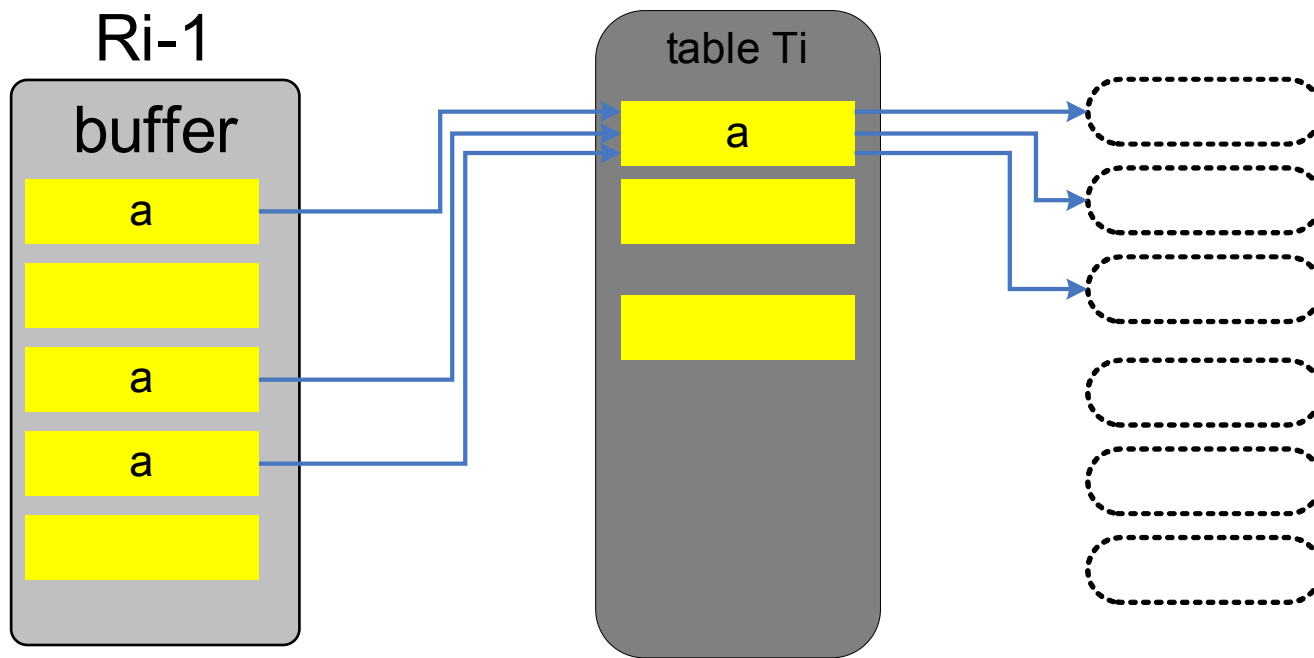
- BKA: access key is part of previous table record



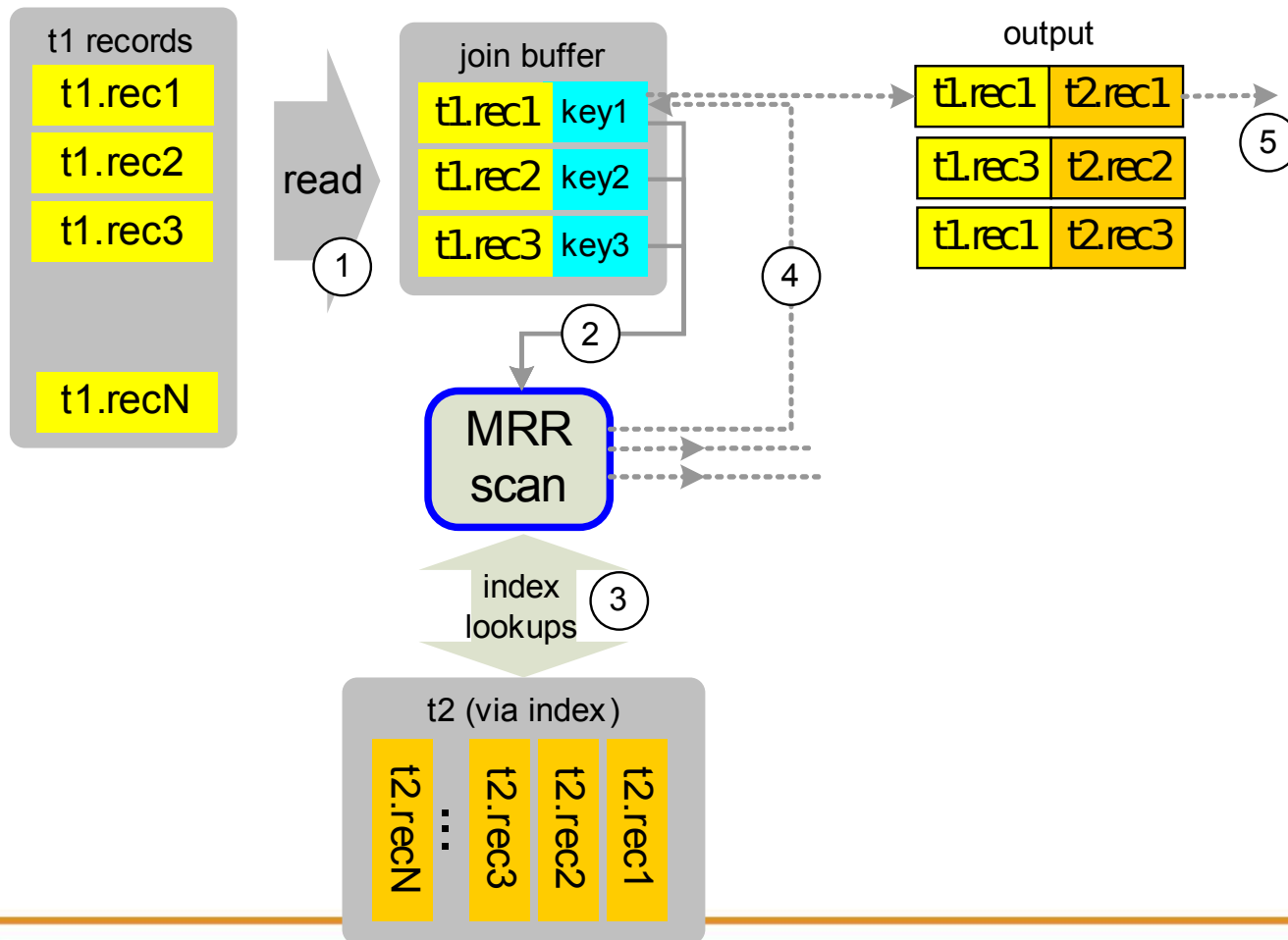
- BKA: access key is not part of previous table record



Batched Key Access Join \leftrightarrow Blocked Nested Loops Join



Batched Key Access: Basic Flowchart



BKA Development Benchmark

2 processor Opteron

[AMD Opteron(tm) Processor 248.cpu

MHz: 2210.161, cache size: 1024 KB]

3GB of memory

Quite a limited disk space, a slow HDD (no RAID).

Query 1:

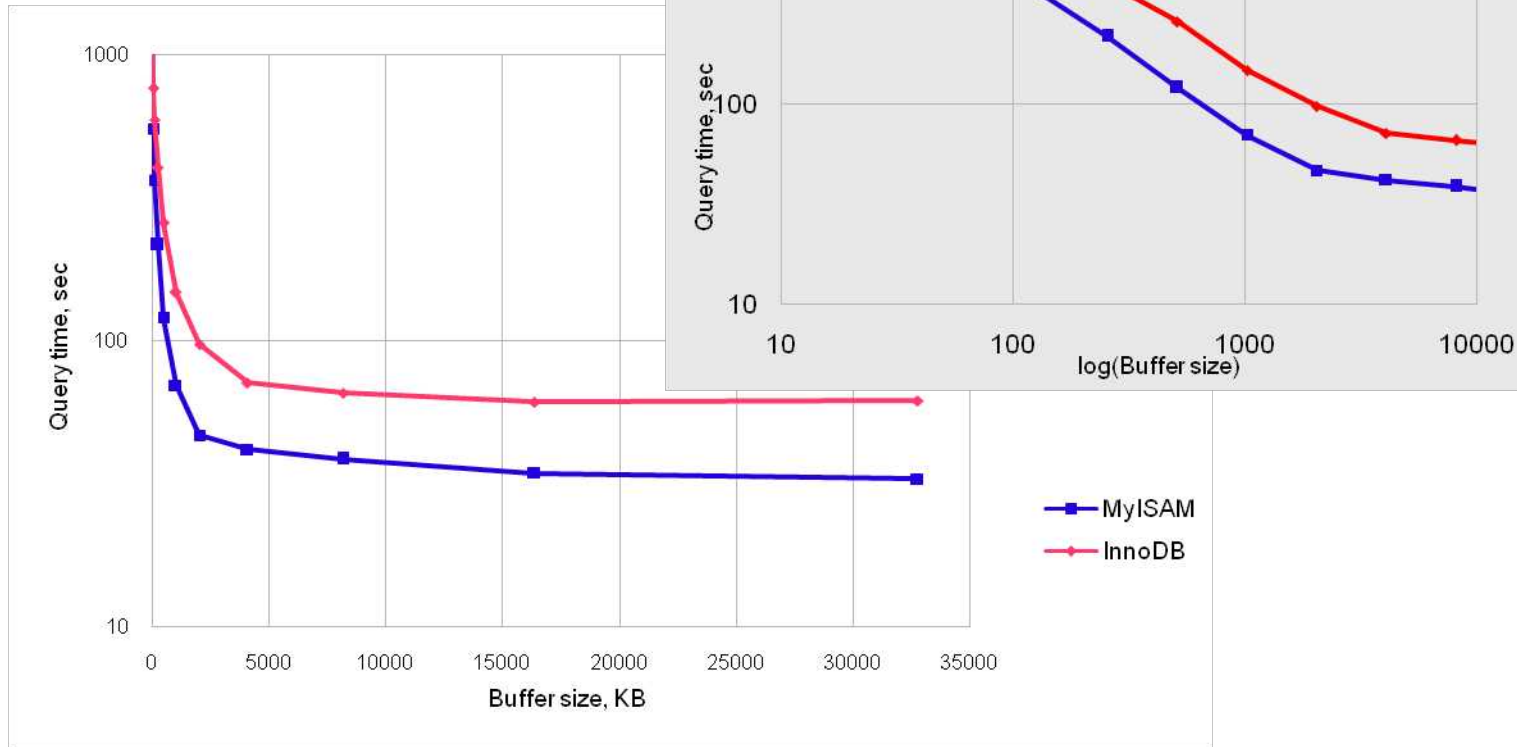
```
SELECT COUNT(*) FROM part, lineitem ON l_partkey=p_partkey
WHERE p_retailprice>?[1525] AND l_discount>?[0.04];
```

Query 2 :

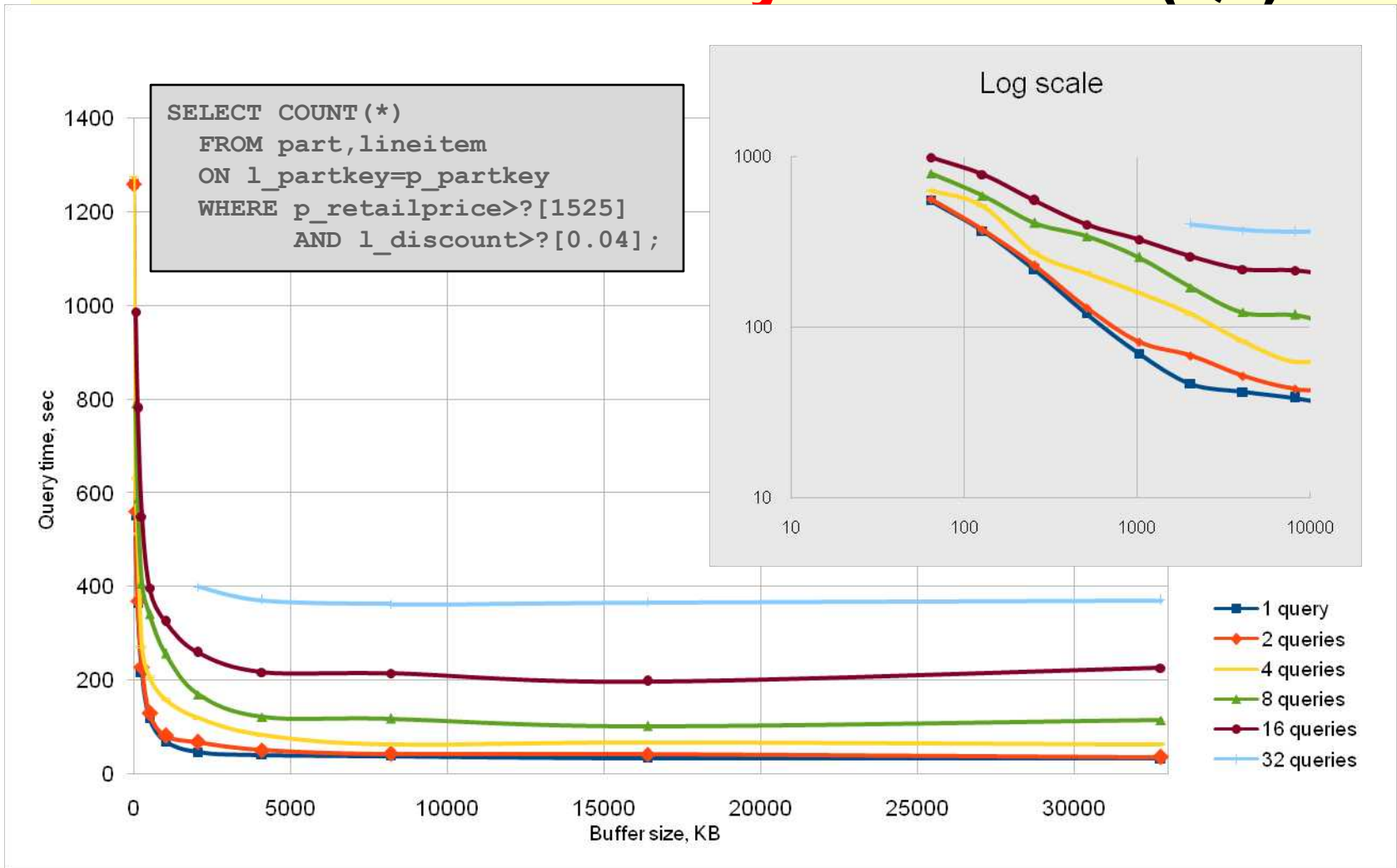
```
SELECT COUNT(*) FROM part, lineitem, partsupp
WHERE l_partkey=p_partkey AND p_retailprice>?[1525] AND
l_discount>?[0.04] AND ps_partkey=p_partkey AND
ps_supplycost>?[520]
```

BKA Benchmark: Speed-up with Join Buffer

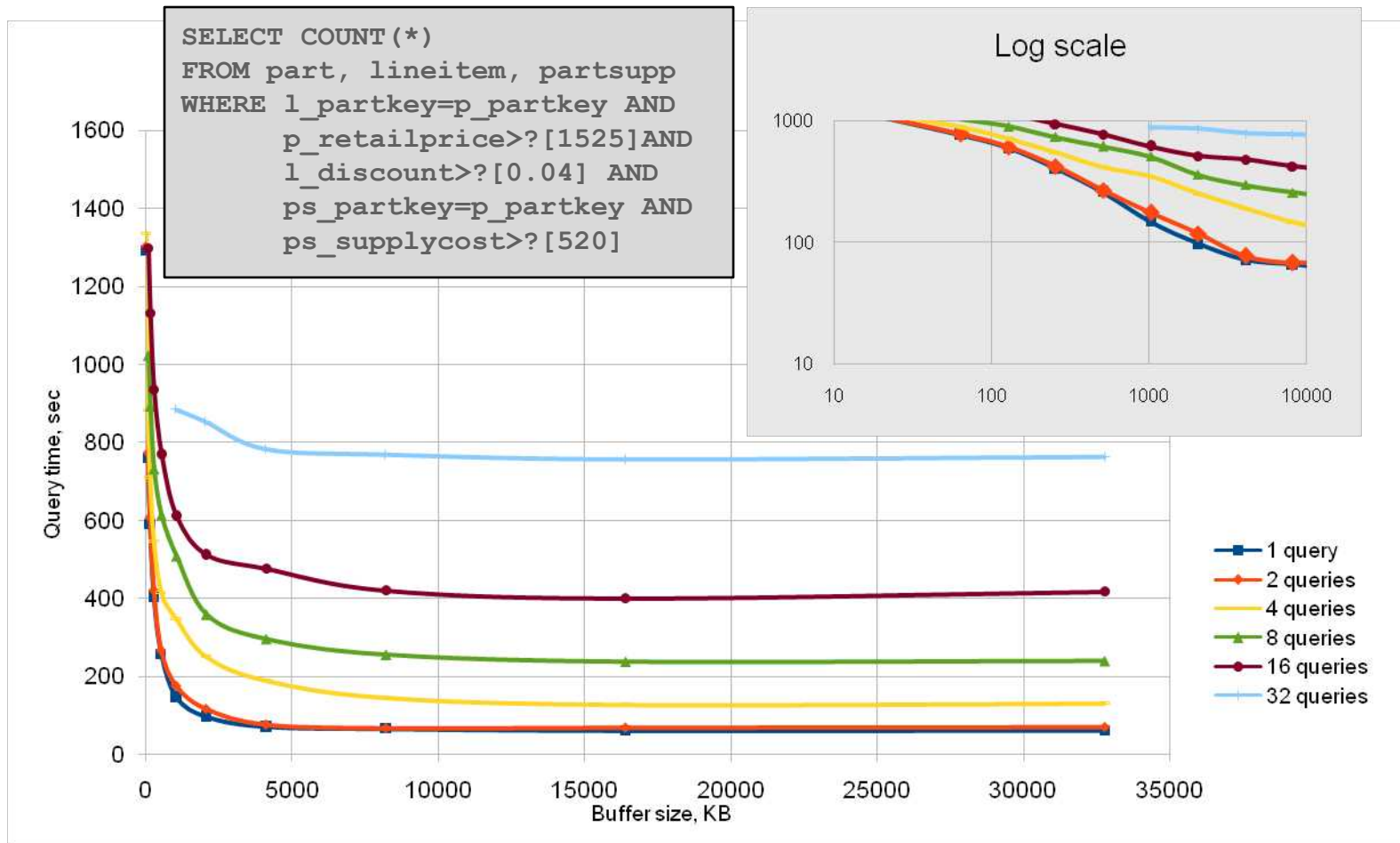
```
SELECT COUNT(*)  
FROM part, lineitem  
ON l_partkey=p_partkey  
WHERE p_retailprice>1525  
AND l_discount>0.04;
```



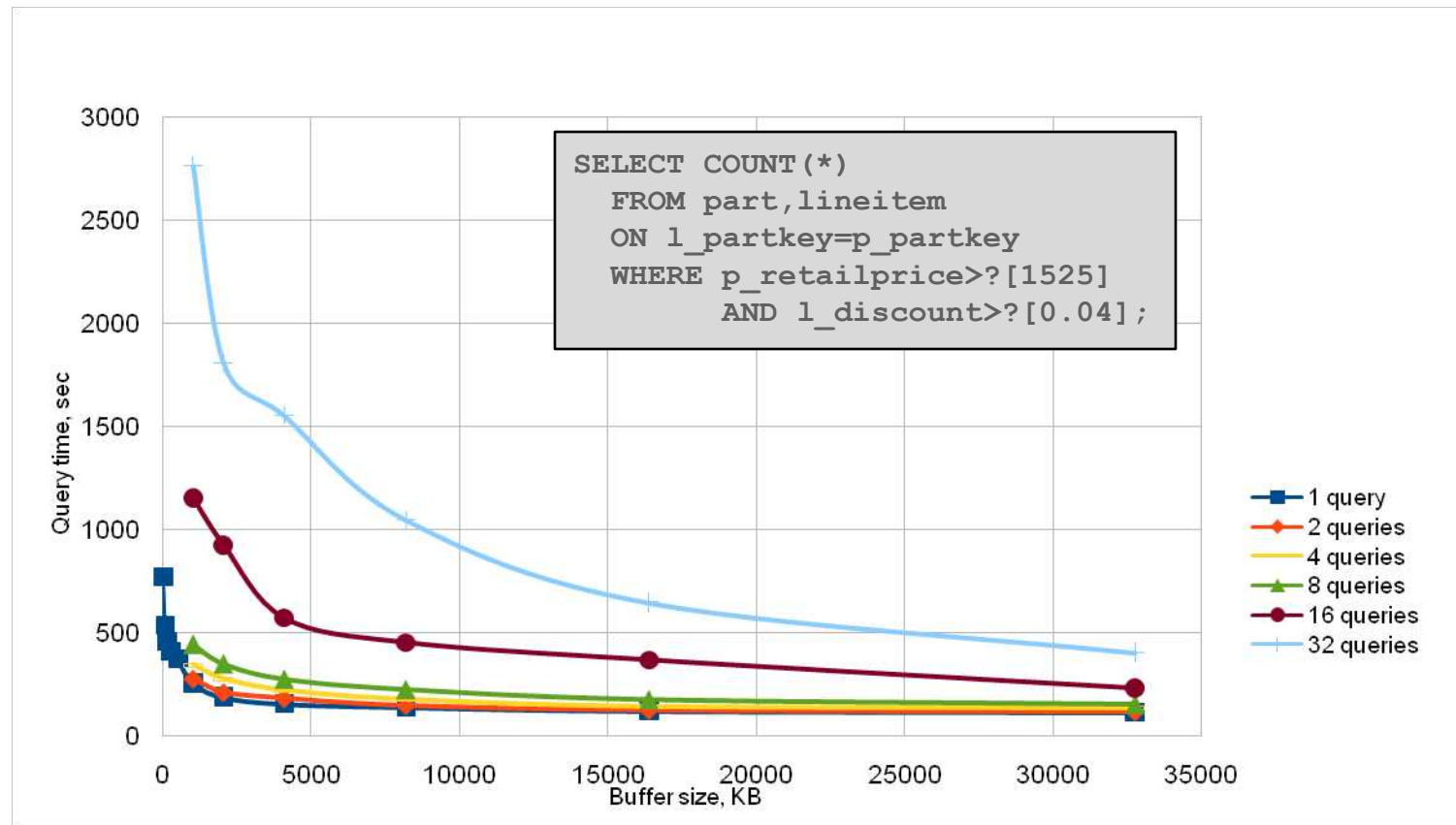
BKA Benchmark: How **MyISAM** scales (Q1)



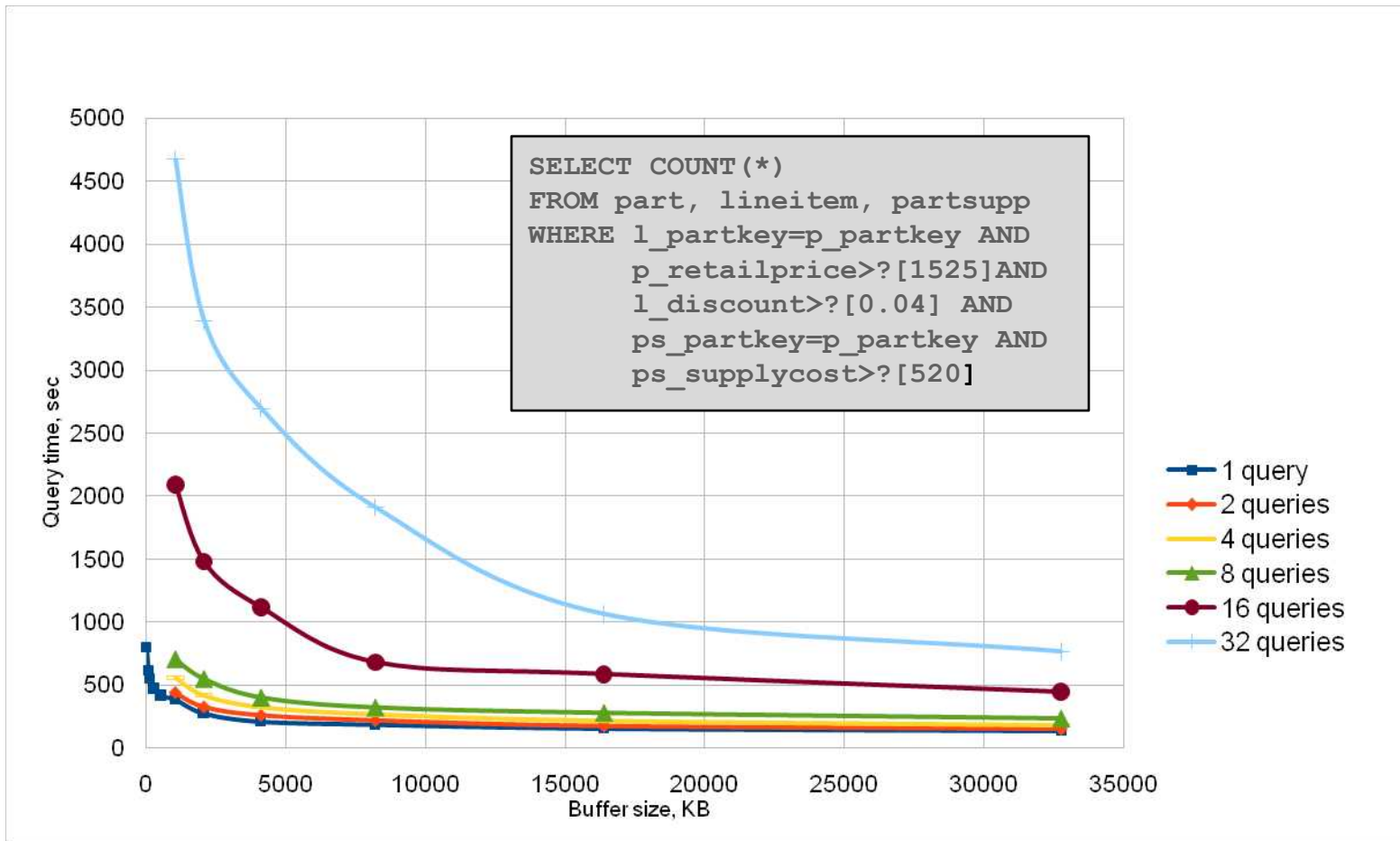
BKA Benchmark: How **MyISAM** scales (Q2)



BKA Benchmark: How **InnoDB** scales (Q1)

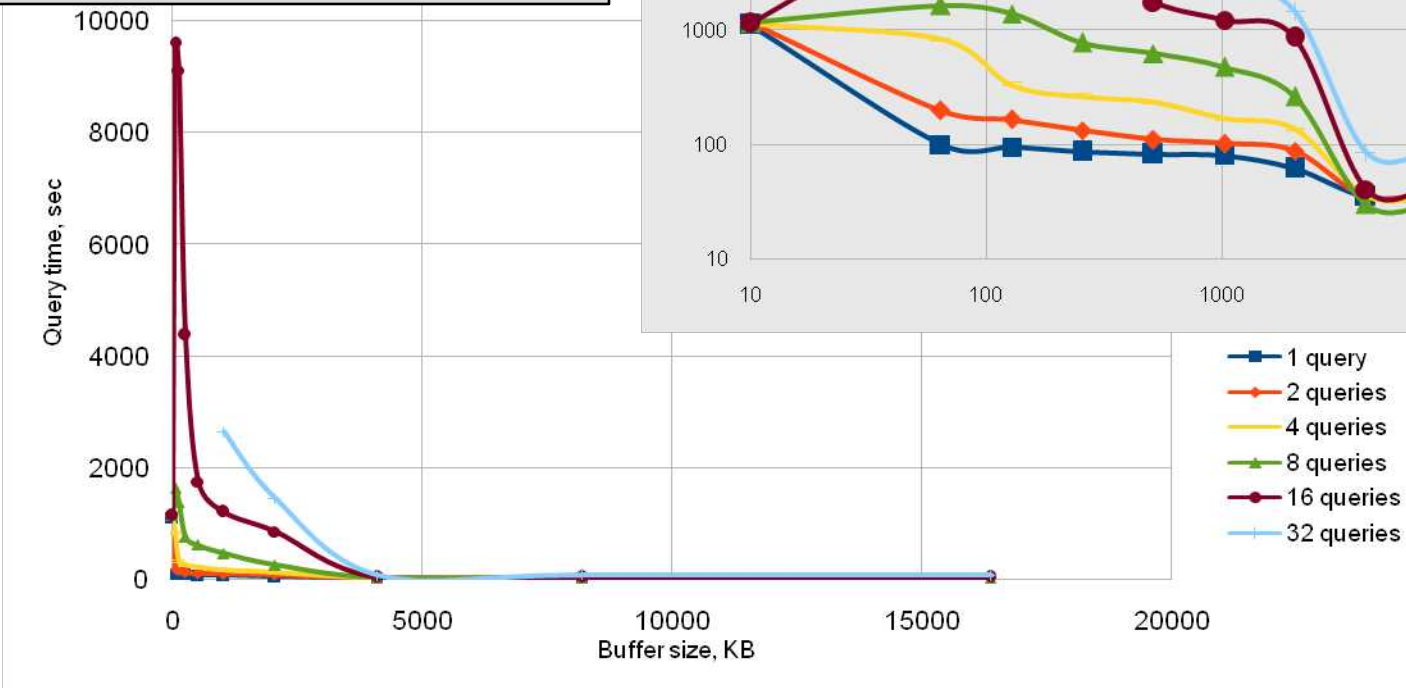


BKA Benchmark: How **InnoDB** scales (Q2)

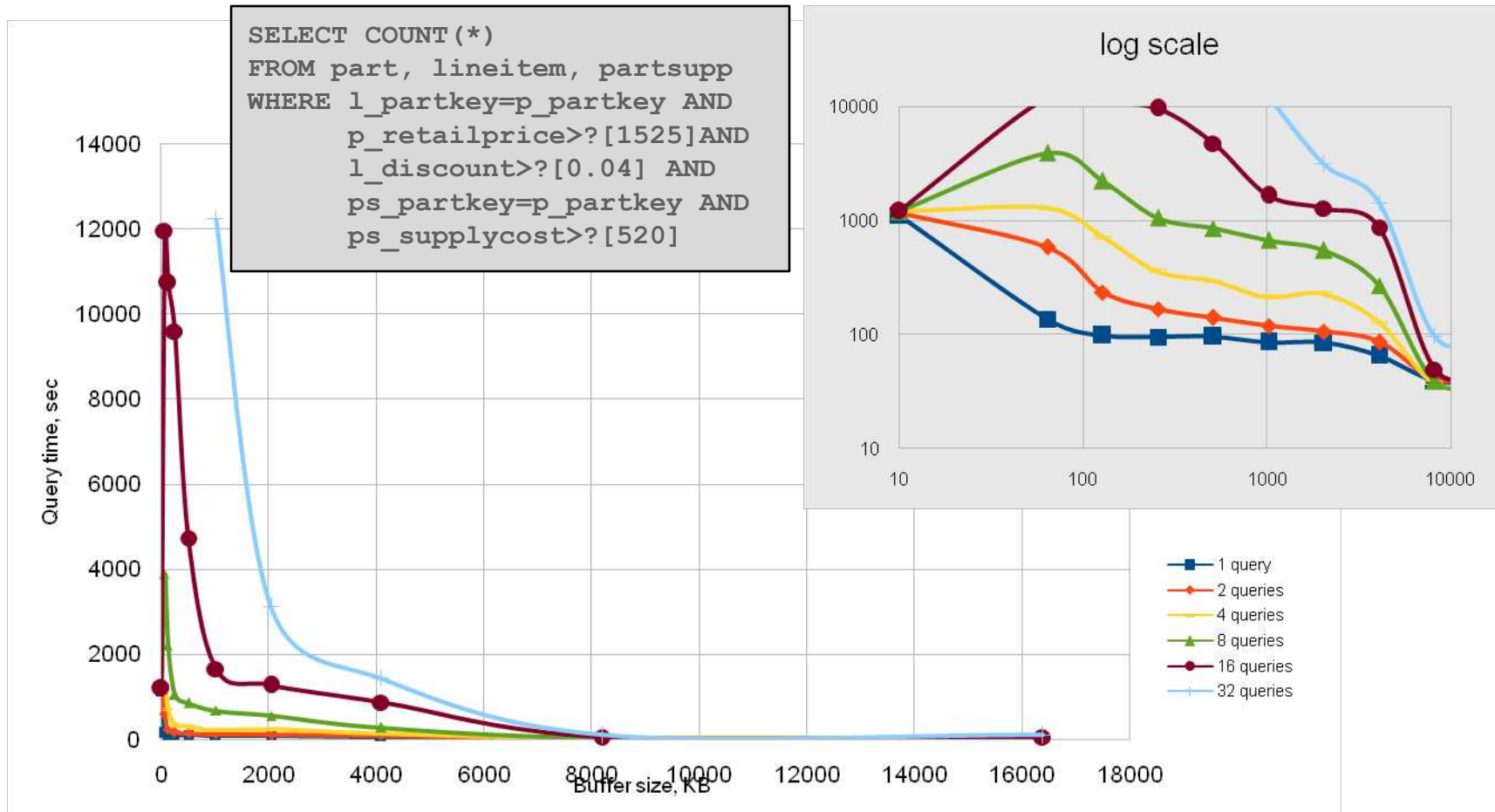


BKA Benchmark: How Hash Join of PostgreSQL scales (Q1)

```
SELECT COUNT(*)  
FROM part,lineitem  
ON l_partkey=p_partkey  
WHERE p_retailprice>?[1525]  
AND l_discount>?[0.04];
```



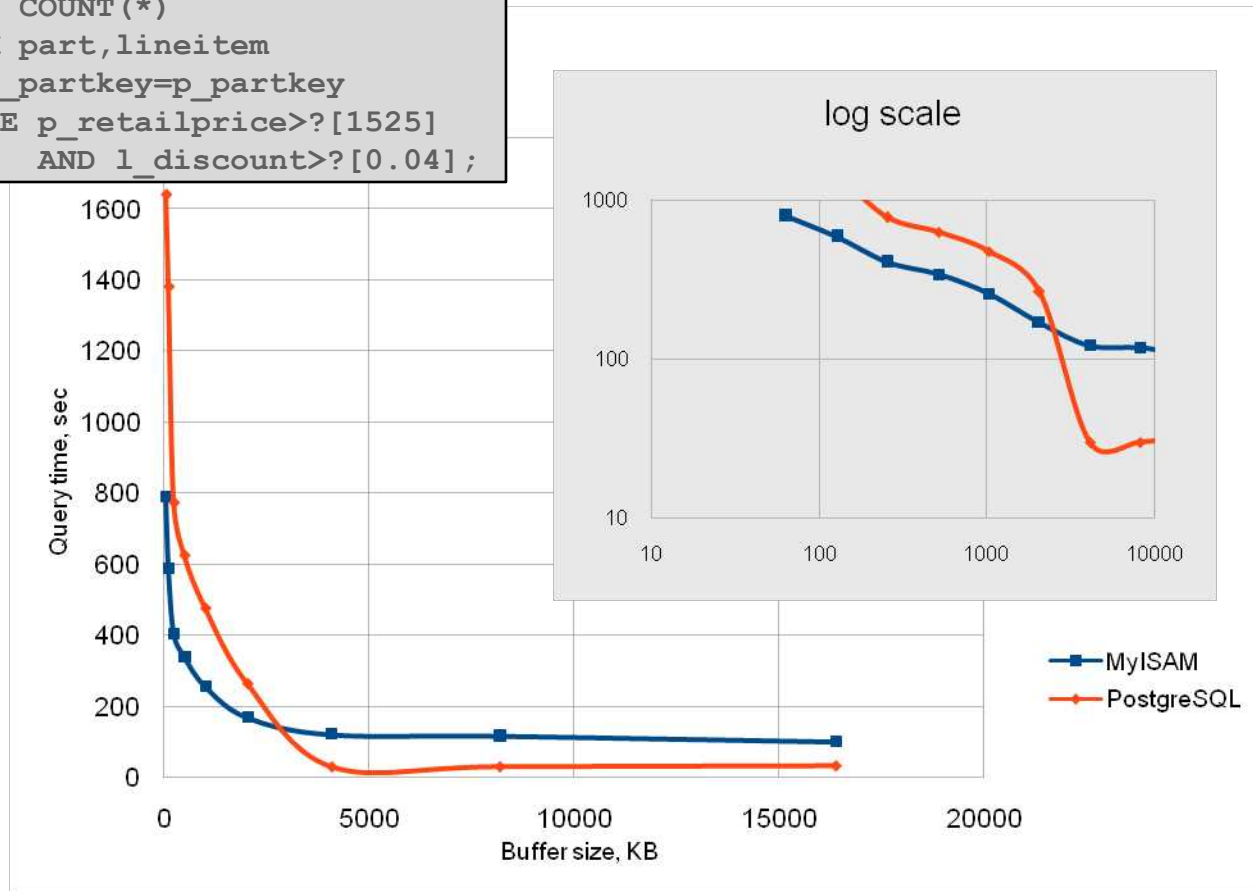
BKA Benchmark: How Hash Join of PostgreSQL scales (Q2)



BKA Benchmark: Q1 x 8

BKA Join (MyISAM) vs Hash Join (PostgreSQL)

```
SELECT COUNT(*)  
FROM part,lineitem  
ON l_partkey=p_partkey  
WHERE p_retailprice>?[1525]  
AND l_discount>?[0.04];
```



Presented by

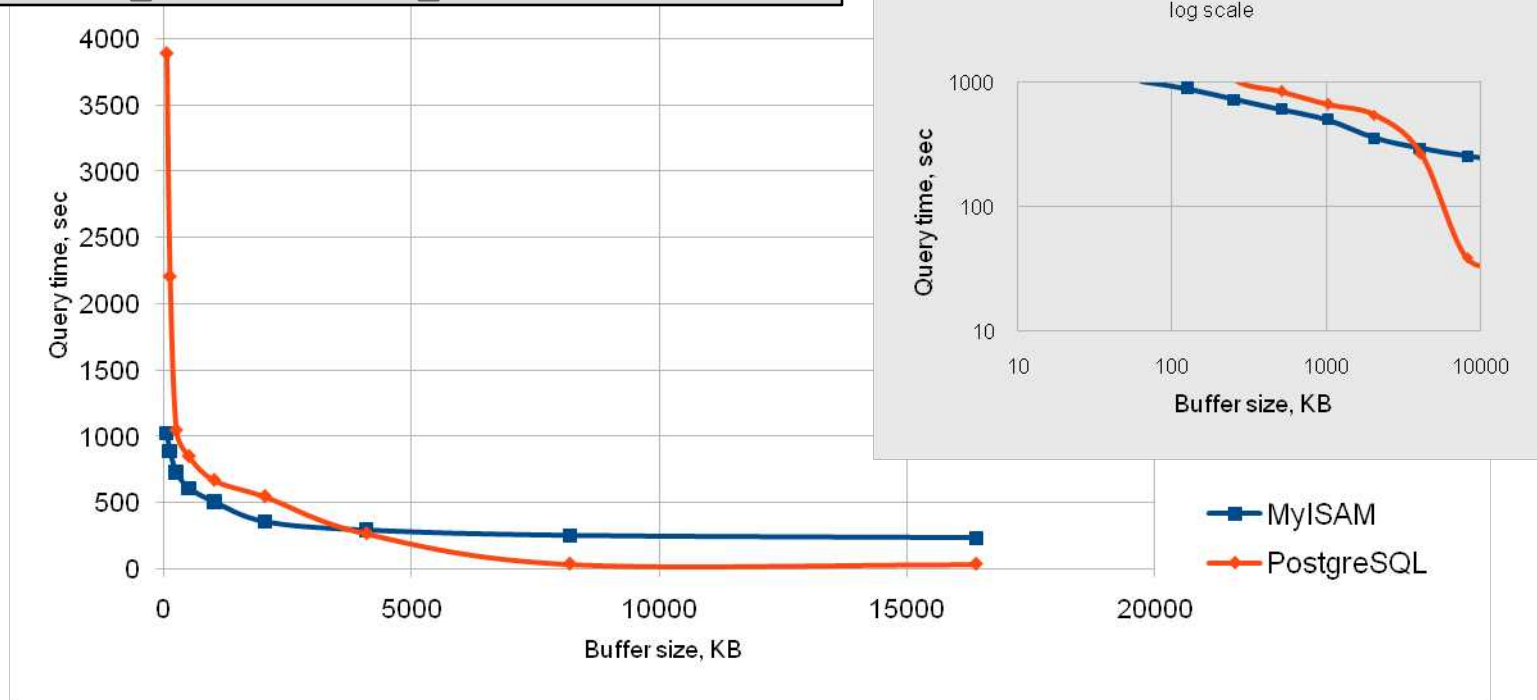


O'REILLY

BKA Benchmark: Q2 x 8

BKA Join (MyISAM) vs Hash Join (PostgreSQL)

```
SELECT COUNT(*)
FROM part, lineitem, partsupp
WHERE l_partkey=p_partkey AND
p_retailprice>?[1525]AND l_discount>?[0.04] AND
ps_partkey=p_partkey AND ps_supplycost>?[520]
```



Batched Key Access: FAQ 1

1. Can BKA be applied only to inner joins?

No, it can be applied to outer joins and semi-joins as well (including nested outer joins and semi-joins with several inner tables).

2. Does BKA employ conditional pushdown of predicates from the *where* clauses of queries with outer joins?

Yes, it does.

3. Does BKA support the *first match* strategy for semi-joins and the *non-exists* strategy for outer joins.

Yes, it does support both.

Batched Key Access: FAQ 2

4. Is the *index condition pushdown* supported by BKA?

Currently only for conditions that can be pushed fully to indexes.

5. For which engines BKA join is always beneficial?

For *remote* engines (like NDB Cluster).

6. In what situations it does not make sense to use BKA joins?

With tables that can be placed entirely in memory,
with join queries that require single lookups into the joined tables.

7. Will BKA/MRR be supported by the Falcon engine?

Hopefully it will be soon.