Центральный научно-исследовательский институт экономики, информатики и систем управления

ЦНИИ ЭИСУ

GIN generalization

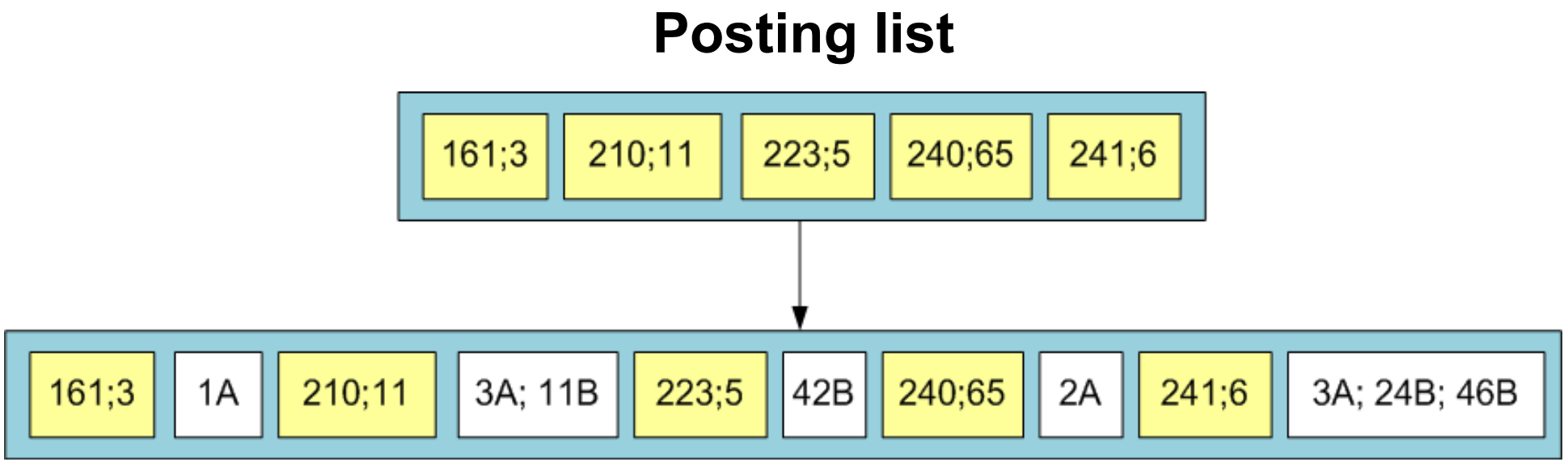
Alexander Korotkov, Oleg Bartunov

Work supported by Federal Unitary Enterprise Scientific-Research Institute of Economics, Informatics and Control Systems

Presentation plan

- New storage (additional infromation + compression)
- Fast scan (skip parts of posting trees)
- Ordering in index
- Planner optimization
- Applications

Additional information storage



Additional information interface


```
void config
(
   GinConfig *config
)

Datum *extractValue
(
   Datum itemValue,
   int32 *nkeys,
   bool **nullFlags,
   Datum *addInfo,
   bool *addInfoIsNull
)
```

```
bool consistent
(
   bool check[],
   StrategyNumber n,
   Datum query,
   int32 nkeys,
   Pointer extra_data[],
   bool *recheck,
   Datum queryKeys[],
   bool nullFlags[],
   Datum addInfo[],
   bool addInfoIsNull[]
)
```

ItemPointer


```
typedef struct BlockIdData
{
    uint16    bi_hi;
    uint16    bi_lo;
} BlockIdData;
typedef struct ItemPointerData
{
    BlockIdData ip_blkid;
    OffsetNumber ip_posid;
}
```

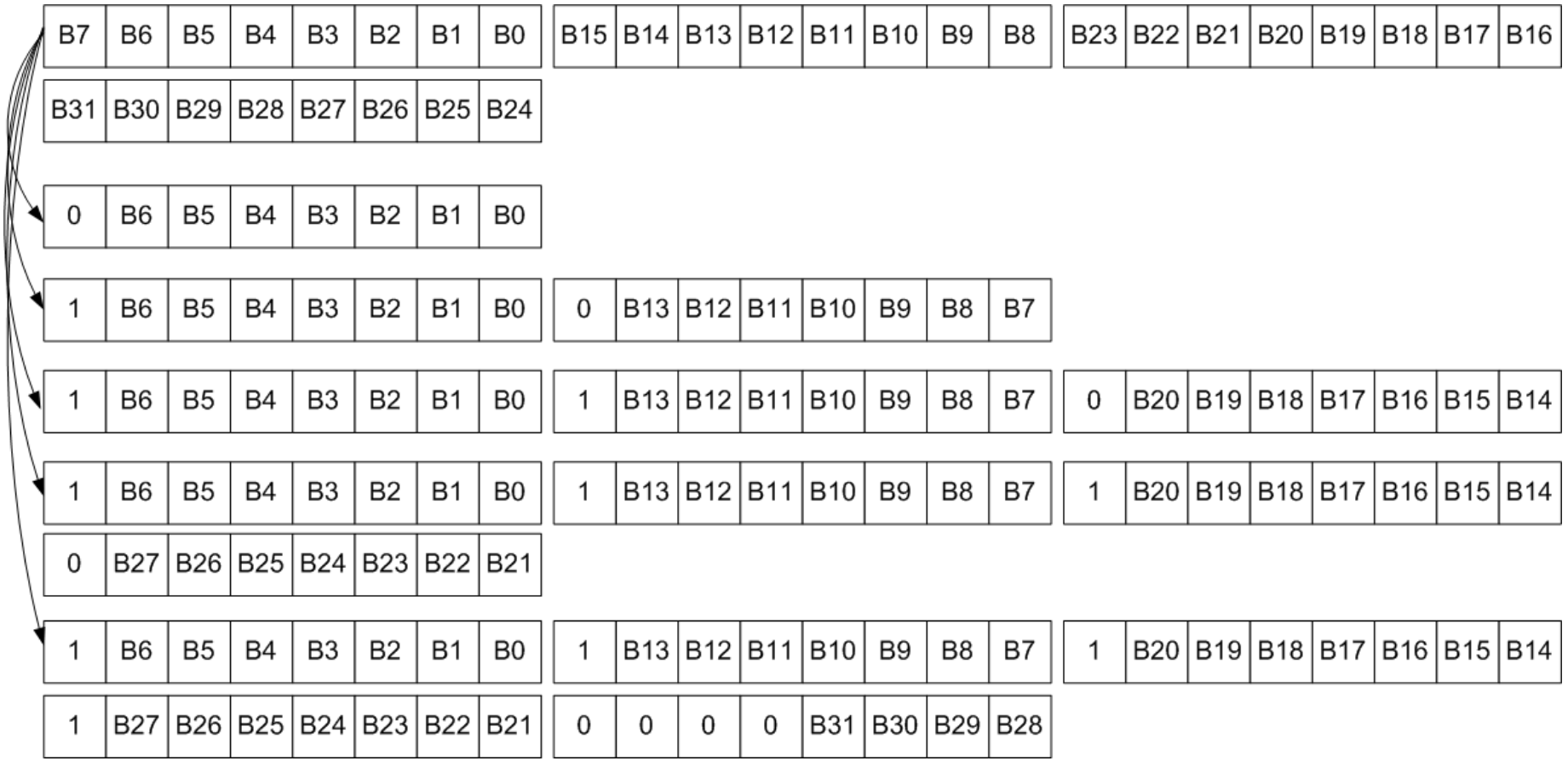
6 bytes!

BlockIdData compression

Центральный научно-исследовательский институт экономики, информатики и систем управления

ЦНИИ ЭИСУ

BlockNumber delta is a small number: use varbyte encoding

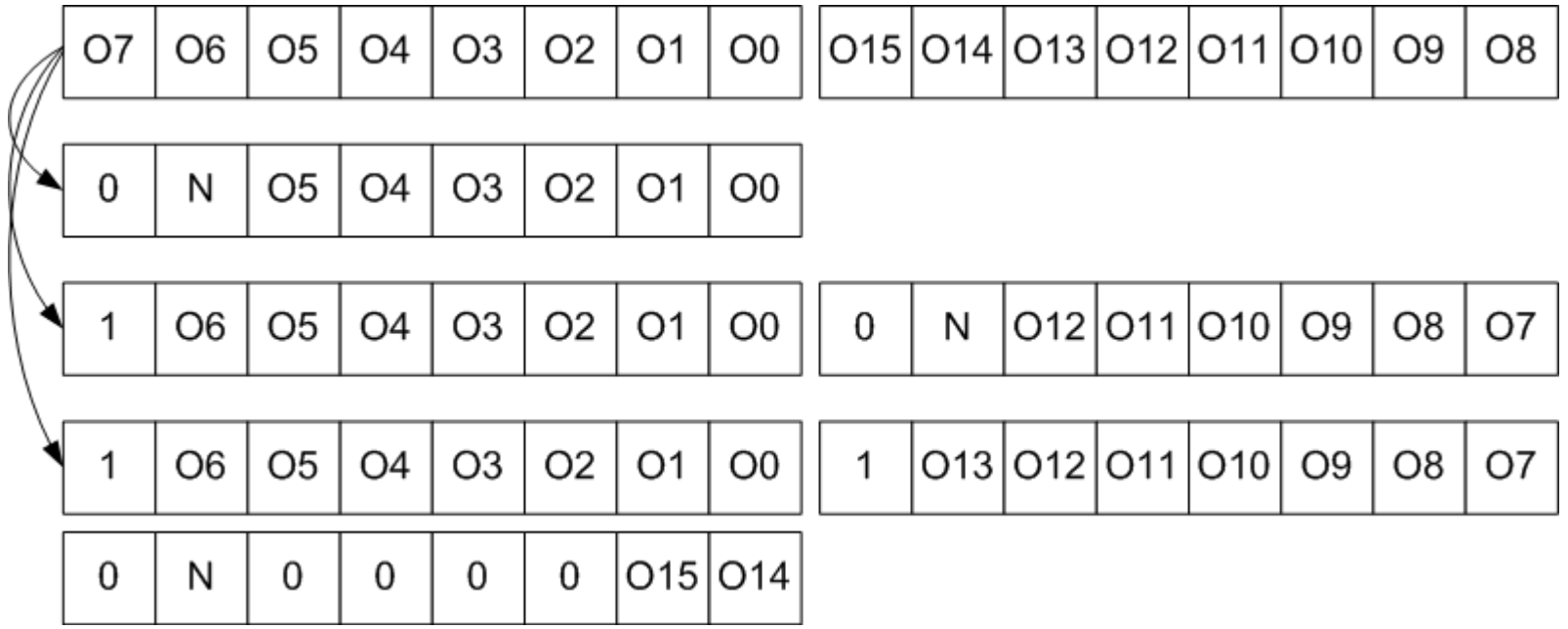


OffsetNumber compression

Центральный научно-исследовательский институт экономики, информатики и систем управления

ЦНИИЭИСУ

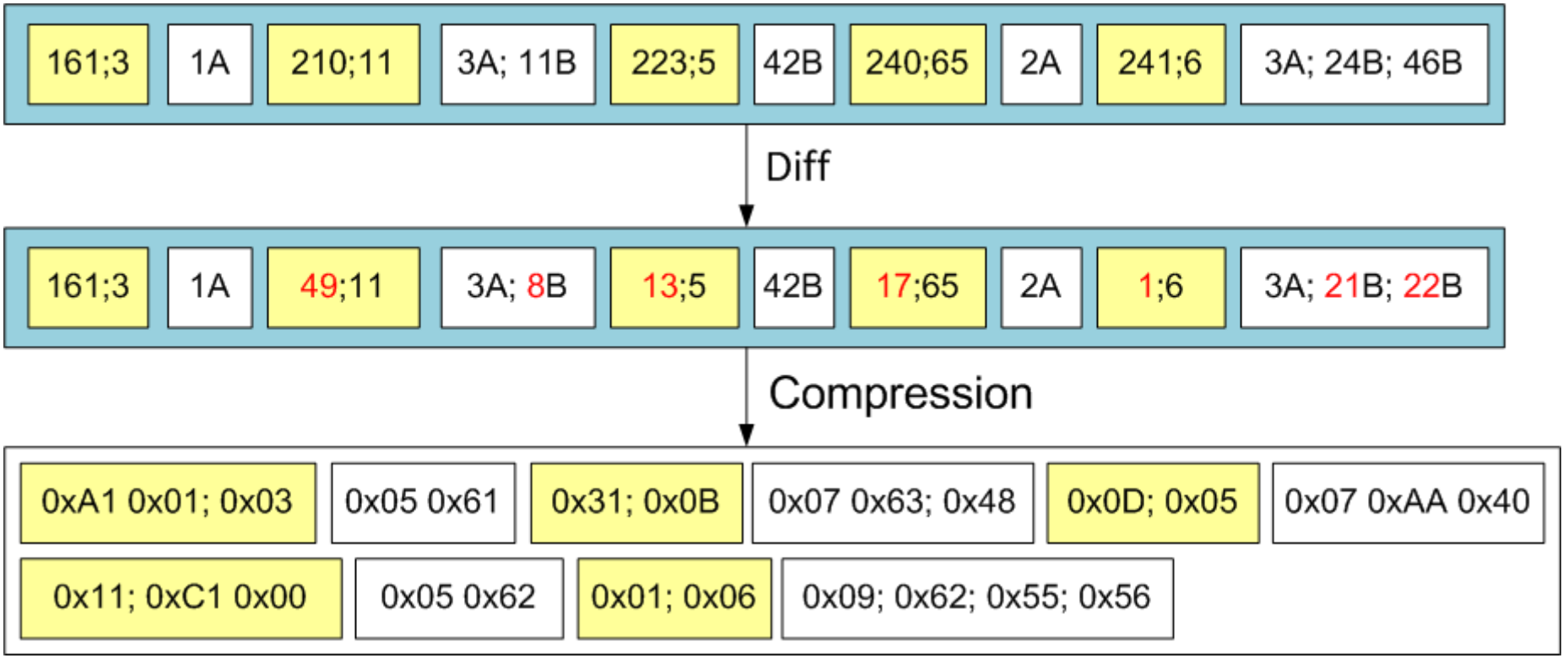
Offset is typically low: use varbyte encoding



O0-O15 – OffsetNumber bits

N – Additional information NULL bit

New storage example



GIN partial match

Without additional information partial match beaves so:

- Save ItemPointers of all matching entries into **bitmap**
- Use bitmap **instead** of partial match entry

GIN partial match with additional information

Join additional information for all entries of each ItemPointer together?

```
Datum joinAddInfo
(
   Datum addInfos[]
)
```

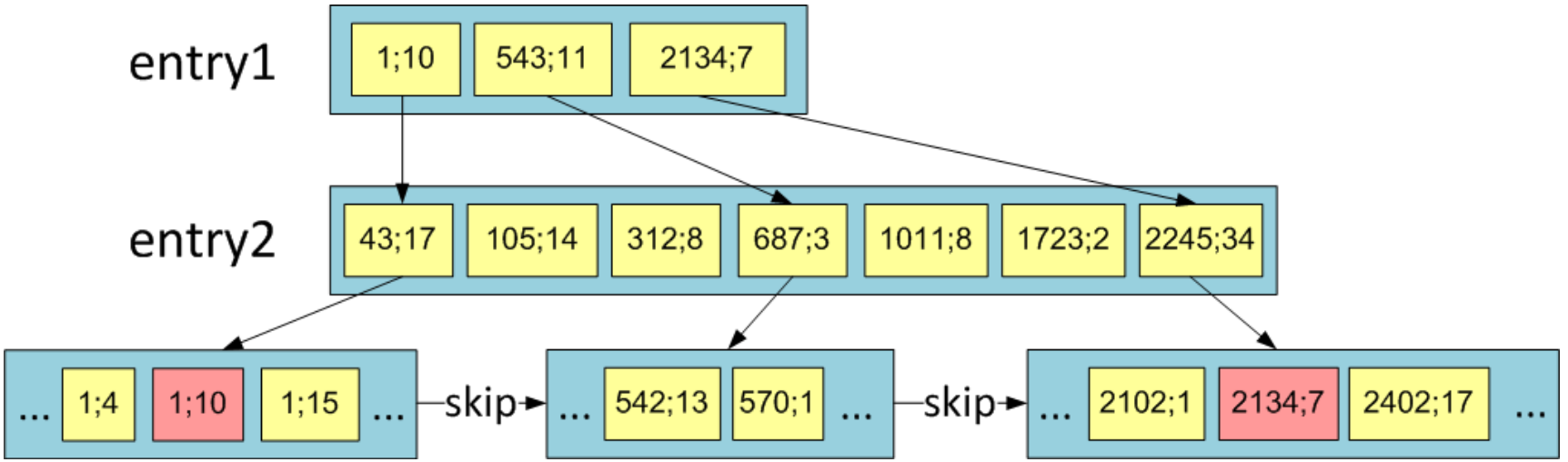
Where to store it?
It might be much more huge than bitmap.

Fast scan

Центральный научно-исследовательский институт экономики, информатики и систем управления

ЦНИИ ЭИСУ

entry1 && entry2



Fast scan: interface

preConsistent: consistent which allows false positives

```
bool preCconsistent
(
    bool check[],
    StrategyNumber n,
    Datum query,
    int32 nkeys,
    Pointer extra_data[]
)
```

Sorting in index

Uses KNN infrastructure. Woks so:

1. Scan + calc rank
2. Sort
3. Return using gingettuple one by one

It is right design at all? Should we do sorting in a separate node?

Sorting in index: interface


```
float8 calcRank
(
    bool check[],
    StrategyNumber n,
    Datum query,
    int32 nkeys,
    Pointer extra_data[],
    bool *recheck,
    Datum queryKeys[],
    bool nullFlags[],
    Datum addInfo[],
    bool addInfoIsNull[]
)
```

Planner optimization: before


```
test=# EXPLAIN (ANALYZE, VERBOSE) SELECT * FROM test ORDER BY
slow_func(x,y) LIMIT 10;
                                                                QUERY
PLAN

------------------------------------------------------------------------
----------------------------------------
 Limit  (cost=0.00..3.09 rows=10 width=16) (actual time=11.344..103.443
rows=10 loops=1)
   Output: x, y, (slow_func(x, y))
   ->  Index Scan using test_idx on public.test  (cost=0.00..309.25
rows=1000 width=16) (actual time=11.341..103.422 rows=10 loops=1)
         Output: x, y, slow_func(x, y)
 Total runtime: 103.524 ms
(5 rows)
```

Planner optimization: after


```
test=# EXPLAIN (ANALYZE, VERBOSE) SELECT * FROM test ORDER BY
slow_func(x,y) LIMIT 10;
                                                                QUERY PLAN


-------------------------------------------------------------------------
-----------------------------------------
 Limit  (cost=0.00..3.09 rows=10 width=16) (actual time=0.062..0.093
rows=10 loops=1)
   Output: x, y
   ->  Index Scan using test_idx on public.test  (cost=0.00..309.25
rows=1000 width=16) (actual time=0.058..0.085 rows=10 loops=1)
         Output: x, y
 Total runtime: 0.164 ms
(5 rows)
```

Applications

- Better FTS
- Array similarity search (better smlr)
- Positioned n-grams (better pg_trgm)
- Index on regexes (inverted task)

Better FTS

- tsvector >< tsquery = 1/ts_rank(tsvector, tsquery)
- gin_tsvector_config — set bytea additional information type
- gin_extract_tsquery — extract compressed word positions as additional information
- gin_tsquery_pre_consistent — evaluate tsquery ignoring NOTs
- gin_tsquery_relevance — calculate ><

Better FTS: query


```
SELECT
    itemid, title
FROM
    items
WHERE
    fts @@ plainto_tsquery('russian',
                           'квартира арбат')
ORDER BY
    fts >< plainto_tsquery('russian',
                           'квартира арбат')
LIMIT
    10;
```

Better FTS: plan


```
Limit  (cost=40.00..80.22 rows=10 width=400) (actual time=1.559..1.5
  Buffers: shared hit=1236
  ->  Index Scan using fts_idx on items  (cost=40.00..7425.31 rows=1
        Index Cond: (fts @@ '''квартир'' & ''арбат'''::tsquery)
        Order By: (fts >< '''квартир'' & ''арбат'''::tsquery)
        Buffers: shared hit=1236
Total runtime: 1.579 ms
```

Avito.ru: testing

	Without patch	With patch	With patch without tsvector	Sphinx
Table size	6.0 GB	6.0 GB	2.87 GB	-
Index size	1.29 GB	1.27 GB	1.27 GB	1.12 GB
Index build time	216 sec	303 sec	718sec	180 sec*
Queries in 8 hours	**3,0 M**	**42.7 M**	**42.7 M**	**32.0 M**

It flies!

Центральный научно-исследовательский институт экономики, информатики и систем управления

ЦНИИЭИСУ

Thank you for attention!
Questions?