Firebird Configuration, episode 3: InlineSortThreshold and details of sortings

Alexey Kovyazin, President of Firebird Foundation

What operations do require sorting?

- ORDER BY
- GROUP BY
- DISTINCT
- UNION (without UNION ALL)

Sorting example

- SELECT ID, NAME FROM TABLE1 ORDER BY DATE1

34	John	06-JUL-1977
2	Alex	12-MAR-1980
51	Tom	07-DEC-1982
80	Roland	16-MAY-1992
7	Helen	26-NOV-1996
19	Nataly	03-APR-1974
39	Carlos	20-AUG-1987

7	Helen	26-NOV-1996
80	Roland	16-MAY-1992
39	Carlos	20-AUG-1987
51	Tom	07-DEC-1982
2	Alex	12-MAR-1980
34	John	06-JUL-1977
19	Nataly	03-APR-1974

How Sorting Works

$$\text{SELECT } \underbrace{\text{ID, NAME}}_{\text{RESULT OF SORTING}} \text{ FROM TABLE1 ORDER BY } \underbrace{\text{DATE1}}_{}$$

- There are 3 ways how sorting can work:
  - SORT
  - REFETCH/INLINE
  - INDEX

SORT

SELECT ID, NAME FROM TABLE1 ORDER BY DATE1

RESULT OF SORTING

KEY OF SORTING

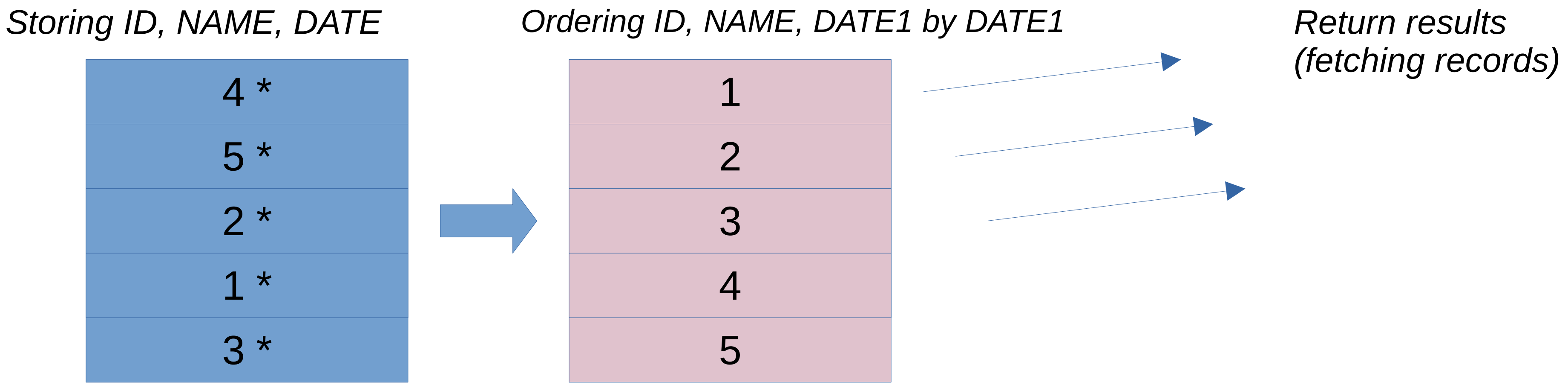
ID INTEGER (8 bytes)	NAME VARCHAR(100)	DATE 8 bytes

116 bytes

SORT

3 Steps
1. Read both result columns and key into the single wide record set, stored in the temp area and, if not enough, on disk
2. Sort recordset by key columns
3. Return results according the fetch command

SELECT ID, NAME
FROM TABLE1 ORDER BY DATE1


Storing ID, NAME, DATE
4 *
5 *
2 *
1 *
3 *
Ordering ID, NAME, DATE1 by DATE1
1
2
3
4
5
Return results (fetching records)

SORT with *

SELECT ID, NAME FROM TABLE1 ORDER BY DATE1

Record length

KEY OF SORTING

ID INTEGER (8 bytes)	NAME VARCHAR(100)	DATE 8 bytes

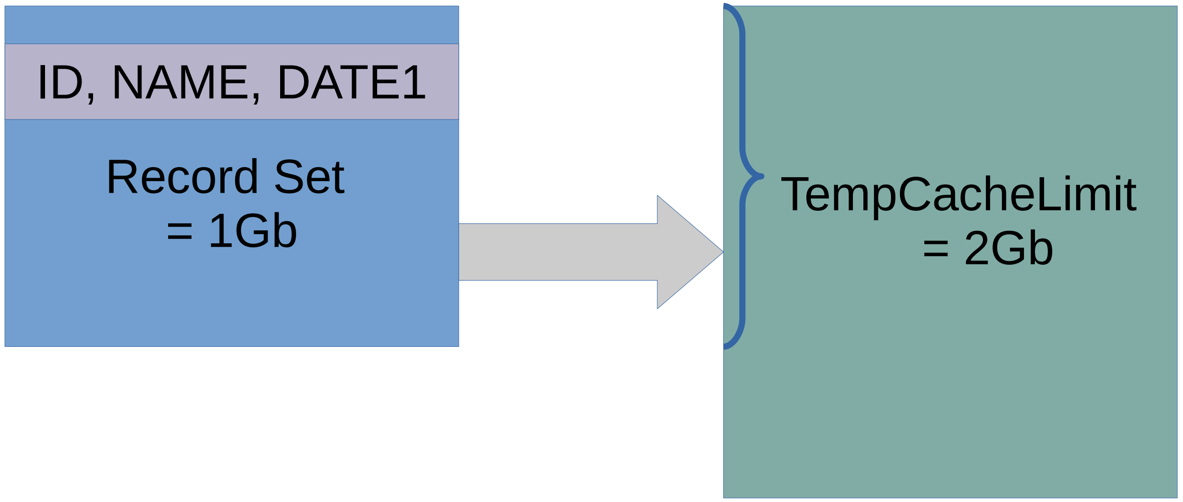
Total 116 bytes

SELECT * FROM TABLE1 ORDER BY DATE1

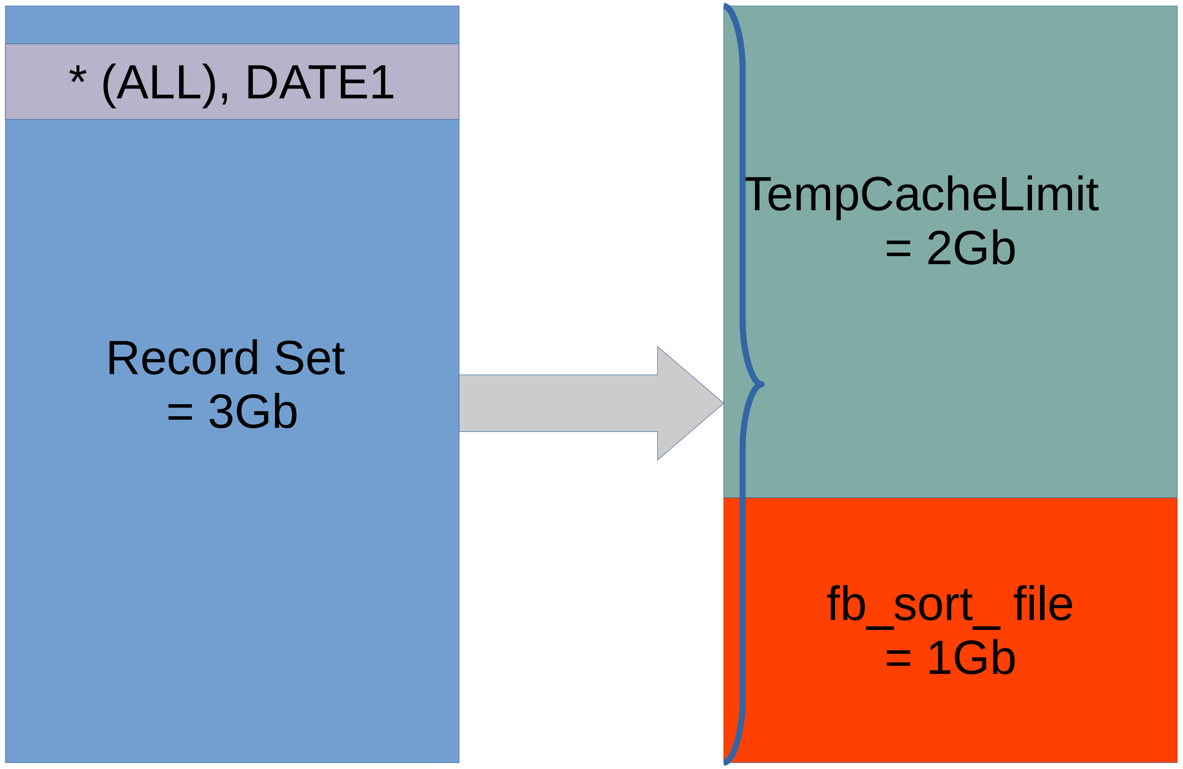
* = many fields e.g. 20000 bytes	DATE 8 bytes

Total 20008 bytes

SORT — small sorting



SORT — big sorting



How can we know the size of sorting?

- In isql
  - set explain;
- In trace
  - explain_plan = true

Examples

```
SQL> show table rdb$types;

RDB$FIELD_NAME  (RDB$FIELD_NAME) CHAR(63) CHARACTER SET UTF8 Nullable

RDB$TYPE        (RDB$GENERIC_TYPE) SMALLINT Nullable

RDB$TYPE_NAME   (RDB$TYPE_NAME) CHAR(63) CHARACTER SET UTF8 Nullable

RDB$DESCRIPTION (RDB$DESCRIPTION) BLOB segment 80,

       subtype TEXT CHARACTER SET UTF8 Nullable

RDB$SYSTEM_FLAG (RDB$SYSTEM_FLAG) SMALLINT Not Null
```

Example — sorting records 284 bytes

```
SQL> select rdb$type_name from rdb$types order by
rdb$type;

Select Expression

    -> Sort (record length: 284, key length: 8)

        -> Table "RDB$TYPES" Full Scan
```

Example — sorting record 548 bytes

```
SQL> select * from rdb$types order by rdb$type;

Select Expression
    -> Sort (record length: 548, key length: 8)
        -> Table "RDB$TYPES" Full Scan
```

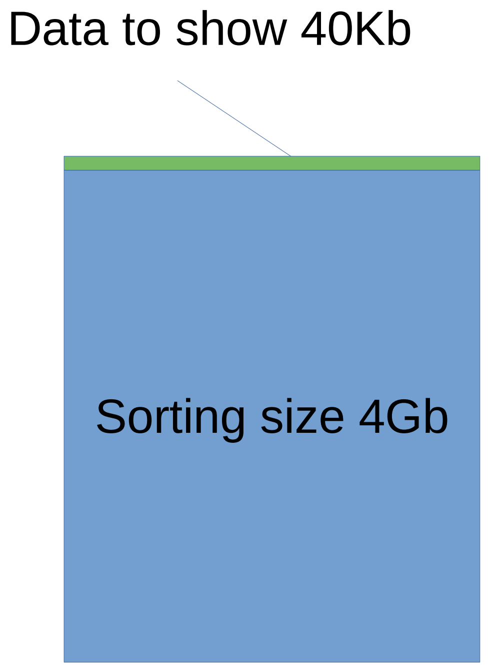
Record size calculation

- Header for each record — always aligned to 32 bytes
  - 8 bytes — transaction’s number
  - 1 byte — DBKEY validity
  - 8 bytes — DBKEY
  - 8 bytes — key for recordset
- Returned data
  - Integer — 8 bytes
  - Timestamp — 8 bytes
  - Varchar (NNN) = NNN * charset (4 for UTF8)

Some calculations

- Sorting 1000000 records, 548 bytes = ~522Mb
- CREATE TABLE T1(i1 integer, v1 varchar(999);
  - SELECT * FROM T1 ORDER BY i1
    - Each record 8b+999*4 (UTF8)+24 bytes = 4028 bytes
    - 1mln records = ~3,7Gb

Example



SELECT FIRST 10 * FROM T1 ORDER BY F1

- Table size = 1000000 records
- Width of record to sort (ALL + F1) = 4K
- F1 size = 8 bytes

To show 10 first records, Firebird will sort 4Gb of data!

REFETCH

- In firebird.conf of Firebird 4
  - InlineSortThreshold=NN #bytes
- In firebird.conf of HQbird 3
  - SortDataStorageThreshold=NN #bytes

- If optimizer sees that record is bigger than specified limit, it will sort only keys, and re-read records by the key

Example Refetch

```
SQL> create table TEST11(I1 integer, vbig varchar(20000));
SQL> show table TEST11;
I1          INTEGER Nullable
VBIG        VARCHAR(20000) Nullable

SQL> select * from TEST11 ORDER BY I1;
Select Expression
    -> Refetch
        -> Sort (record length: 28, key length: 8)
            -> Table "TEST11" Full Scan
```

How refetch works

SELECT * FROM TEST11 ORDER BY I1

1. Read records, build keys for records set
2. Sort keys as usual
3. Re-read records according the sorted keys
4. Return results

SELECT * FROM T1

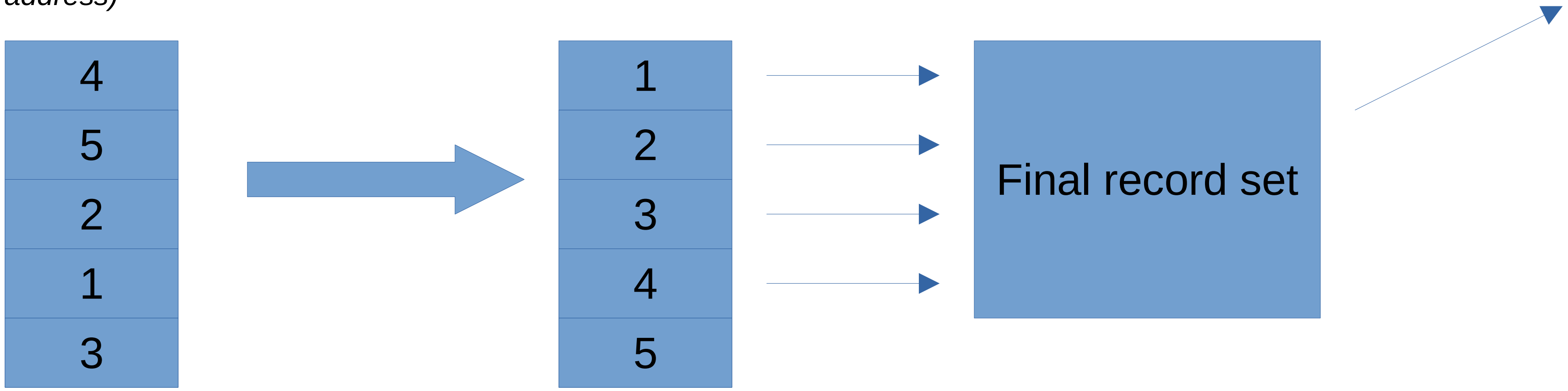
Reading records, building keys (I1+address)

ORDER BY I1

Ordering keys

Re-reading records according the sorted keys

Return results



ORDER INDEX

- If there is index matching the condition in ORDER BY or GROUP BY, Firebird will try to use index (with some limitation)
  - In this case Firebird will not use temp area to store records

Why ORDER INDEX is not always great

- It can be used to order only one table
- Access through index usually requires 3-4x more fetches operations than NATURAL fetching and therefore slower for sorting large amounts of data!

Comparsion

	SORT	REFETCH	INDEX
Use Temp area?	Yes	Yes	No
Sort record and key?	Yes	No	No
Good to sort millions of records with small or moderate size?	Yes, will	Depends	No
Good to sort millions of records with extra large size (>16K)?	No	Yes	No
Good to return only a few (<1000) first rows?	No	Depends	Yes
Order/Group condition can have columns from various tables?	Yes	Yes	No
Reads tables twice?	No	Yes	No

Recommendations for tuning InlineSortThreshold

- Find queries with SORT plan generating large sort files
- Set InlineSortThreshold less than their records length
- Monitor production

Analysing actual temp space usage

- Unfortunately, in the current versions of vanilla Firebird there is no monitoring of used space in TempCacheLimit
  - HQbird has TempSpaceThreshold parameter to report operations which use more space than set

Questions?

- ak@firebirdsql.org
-

Allocating and deallocating temp space

- TempCacheLimit sets the limit, the memory is allocated on demand.
- Temp space is allocated by blocks, specified by parameter TempBlockSize, default =1Mb
- Temp space is deallocated when query finishes the fetch — i.e., when all records of resultset are returned to the client