Optimizations

Improved PLAN clause

D. Yemanov

A PLAN clause optionally allows you to provide your own instructions to the engine and have it ignore the plan supplied by the optimizer. Firebird 2 enhancements allow you to specify more possible paths for the engine.

For example:

```
PLAN (A ORDER IDX1 INDEX (IDX2, IDX3))
```

For more details, please refer to the topic in the DML section, Query plans, Improvements in handling user-specified query plans.

Optimizer improvements

This chapter represents a collection of changes done in Firebird 2.0 to optimize many aspects of performance.

For all databases

The following changes affect all databases.

Some general improvements

O. Loa, D. Yemanov

- Much faster algorithms to process the dirty pages tree.

Firebird 2.0 offers a more efficient processing of the list of modified pages, a.k.a. the dirty pages tree. It affects all kinds of batch data modifications performed in a single transaction and eliminates the known issues with performance getting slower when using a buffer cache of >10K pages. This change also improves the overall performance of data modifications.

- Increased maximum page cache size to 128K pages (2GB for 16K page size).

Faster evaluation of IN() and OR

O. Loa

Constant IN predicate or multiple OR Booleans are now evaluated faster.

Sparse bitmap operations were optimized to handle multiple OR Booleans or an IN (<constant list>) predicate more efficiently, improving performance of these operations.

Improved UNIQUE retrieval

A. Brinkman

The optimizer will now use a more realistic cost value for unique retrieval.

More optimization of NOT conditions

D. Yemanov

NOT conditions are simplified and optimized via an index when possible.

Example

(NOT NOT A = 0) → (A = 0) (NOT A > 0) → (A ⇐ 0)

Distribute HAVING conjunctions to the WHERE clause

If a HAVING clause or any outer-level SELECT refers to a field being grouped by, this conjunct is distributed deeper in the execution path than the grouping, thus allowing an index scan to be used. In other words, it allows the HAVING clause not only be treated as the WHERE clause in this case, but also be optimized the same way.

Examples

```
select rdb$relation_id, count(*)
from rdb$relations
group by rdb$relation_id
having rdb$relation_id > 10

select * from (
   select rdb$relation_id, count(*)
   from rdb$relations
   group by rdb$relation_id
   ) as grp (id, cnt)
where grp.id > 10
```

In both cases, an index scan is performed instead of a full scan.

Distribute UNION conjunctions to the inner streams

Distribute UNION conjunctions to the inner streams when possible.

Improved handling of CROSS JOIN and Merge/SORT

Improved CROSS JOIN and merge/sort handling.

Better choice of join order for mixed inner/outer joins

Let's choose a reasonable join order for intermixed inner and outer joins.

Equality comparison on expressions

MERGE PLAN may now be generated for joins using equality comparison on expressions.

back to top of page

For ODS 11 databases only

This group of optimizations affects databases that were created under Firebird 2.

Segment-level selectivities are used

See Selectivity maintenance per segment in the Indexing chapter.

Better support for IS NULL and STARTING WITH

Previously, IS NULL and STARTING WITH predicates were optimized separately from others, thus causing non-optimal plans in complex ANDed/ORed Boolean expressions. From v2.0 and ODS11, these predicates are optimized in a regular way and hence benefit from all possible optimization strategies.

Matching of both OR and AND nodes to indexes

Complex Boolean expressions consisting of many AND/OR predicates are now entirely mapped to available indices if at all possible. Previously, such complex expressions could be optimized badly.

Better JOIN orders

Cost estimations have been improved in order to improve JOIN orders.

Indexed order enabled for outer joins

It is now possible for indexed order to be utilised for outer joins, i.e. navigational walk.