Basic tips for faster queries in LDP databases

April 7, 2022

Nassib Nassar
Index Data ApS

Overview

- Only needed if your queries are uncomfortably slow.
- We will focus on some of the easiest things you can do to make queries run faster.
- Query optimization and database design will not be covered today.

Indexes

- An index is like a card catalog for a table in a database.
- It is worthwhile to create indexes on some columns in large tables (i.e. having lots of rows).
- Which columns? Any columns that will be used in:
  - `WHERE` *`column`* `...`
  - `JOIN` *`table`* `ON` *`column`* `...`
  - `ORDER BY` *`column`*
- In these cases, without an index, the database system may have to read all of the data in the table.
- To create an index, use the `CREATE INDEX` statement.

Types of indexes

- A "B-tree" index is the default type of index. It helps with WHERE and JOIN clauses having operators such as =, <>, <, >, <=, or >=, and also helps with ORDER BY clauses.

  ```
  CREATE INDEX ON table (column);
  ```

  e.g.: `CREATE INDEX ON instances (instance_hrid);`

- A "GIN" index is another type of index. It helps with LIKE and ILIKE pattern matching operators.

  ```
  CREATE INDEX ON table USING GIN (column gin_trgm_ops);
  ```

  e.g.: `CREATE INDEX ON marc USING GIN (content gin_trgm_ops);`

Vacuum and analyze

- After creating a large table or modifying its contents, it is helpful to vacuum and analyze it.

  `VACUUM ANALYZE *table*;`

  e.g.: `VACUUM ANALYZE instances;`

- This scans the data to collect statistics that will help the database system plan how to execute queries, and also cleans up any updated/deleted rows.

- Create any indexes before running "vacuum analyze."

- The database system will automatically "vacuum analyze" tables after some period of time, but you can run it manually (as shown above) to be sure that it gets done before you use the table.

Temporary tables

- When creating a large table that you only need briefly, use a temporary table.
- Temporary tables are faster to create than normal tables, because the database system does not worry about trying to recover their data in case of a crash.

```
CREATE TEMP TABLE table AS SELECT ...;
```

```
e.g.: CREATE TEMP TABLE t AS
          SELECT * FROM srs_records WHERE state = 'ACTUAL';
```

- When your connection to the database closes, your temporary tables are automatically deleted.
- Note that temporary tables are not visible to other users.
- Add indexes where needed and "vacuum analyze."

Breaking apart large queries

- Queries with subqueries or CTEs involving large tables are not always handled well by the database system.

  ```
  SELECT ... (SELECT ...);
  ```

  ```
  WITH cte AS (SELECT ...) SELECT ... FROM cte ...;
  ```

- It sometimes helps to create the subqueries/CTEs as separate, intermediate tables.
- If the intermediate tables will not be useful for other queries, create them as temporary tables. But if, as with derived tables, they are useful for other queries as well, then create them as regular tables.
- Add indexes where needed and "vacuum analyze."

Use NOT EXISTS instead of NOT IN

- The "NOT IN" operator tends to be very slow with large tables.

  ```
  WHERE column1 NOT IN (SELECT column2 FROM table);
  ```

- Instead, use "NOT EXISTS":

  ```
  WHERE NOT EXISTS (
      SELECT column2 FROM table WHERE column1 = table.column2
  );
  ```

Simple column names

- A column name used with an expression or function will not benefit from an index, unless it matches the expression used when the index was created.

```
WHERE column + expression = ...

WHERE function(column) = ...
```

- Where possible, try to use simple column names on the left side of an operator, and move the expression to the right side.

```
WHERE column = ... – expression

WHERE column = function'(...)
```

System configuration

- Ask your system administrator to:
  - Increase CPU cores, memory, and I/O specs
  - Use the recommended database configuration settings in the LDP Administrator Guide
  - Upgrade to recent versions of LDP software
- LDP 1.x can be scaled by adding read replicas; Metadb uses a different approach to scaling