IF3140 Sistem Basis Data
SQL Performance Tuning

SEMESTER I TAHUN AJARAN 2024/2025

KNOWLEDGE & SOFTWARE ENGINEERING

Sources

- “Database Systems: Design, Implementation, and Management”, ninth edition, by Coronel, Morris, Rob; chapter 11 (Cengage Learning, 2010)

KNOWLEDGE & SOFTWARE ENGINEERING

SQL Performance Tuning

- Most current-generation relational DBMSs perform automatic query optimization at the server end.
- Most SQL performance optimization techniques are DBMS-specific and, therefore, are rarely portable, even across different versions of the same DBMS.
- **Does this mean that you should not worry about how a SQL query is written because the DBMS will always optimize it?**
- No, because there is considerable room for improvement.
- The DBMS uses *general* optimization techniques, rather than focusing on specific techniques dictated by the special circumstances of the query execution.

KNOWLEDGE & SOFTWARE ENGINEERING

SQL Performance Tuning

- A poorly written SQL query can, *and usually will*, bring the database system to its knees from a performance point of view.
- The majority of current database performance problems are related to poorly written SQL code.
- A carefully written query almost always outperforms a poorly written one.

SQL Performance Tuning

- Most recommendations in this material are related to the use of the SELECT statement.

- Recommendation categories:
  - Index tuning (revisited)
  - Conditional expressions
  - Query formulation

KNOWLEDGE & SOFTWARE ENGINEERING

Index Tuning - Revisited

- Indexes are the most important technique used in SQL performance optimization
- General rule of when indexes are likely to be used:
  - When an indexed column appears by itself in a search criteria of a WHERE or HAVING clause.
  - When an indexed column appears by itself in a GROUP BY or ORDER BY clause.
  - When a MAX or MIN function is applied to an indexed column.
  - When the data sparsity on the indexed column is high.

Index Tuning - Revisited

- Some general guidelines for creating and using indexes
  - Create indexes for each single attribute used in a WHERE, HAVING, ORDER BY, or GROUP BY clause
  - Do not use indexes in small tables or tables with low sparsity.
  - Declare primary and foreign keys so the optimizer can use the indexes in join operations.
  - Declare indexes in join columns other than PK or FK

KNOWLEDGE & SOFTWARE ENGINEERING

Index Tuning – Revisited

- You cannot always use an index to improve performance:
  - In some DBMSs, indexes are ignored when you use **functions** in the table attributes
    - But major databases (such as Oracle, SQL Server, and DB2) now support function-based indexes
    - A **function-based index** is an index based on a specific SQL function or expression. Function-based indexes are especially useful when dealing with derived attributes.
      - Example: an index on YEAR(INV_ DATE).
  - Too many indexes will slow down INSERT, UPDATE, and DELETE operations
  - Some query optimizers will choose only one index to be the driving index for a query, even if your query uses conditions in many different indexed columns

KNOWLEDGE & SOFTWARE ENGINEERING

Conditional Expressions

- A conditional expression is normally placed within the WHERE or HAVING clauses of a SQL statement
- A conditional expression restricts the output of a query to only the rows matching the conditional criteria
- Most of the query optimization techniques mentioned next are designed to make the optimizer's work easier.

Conditional Expressions

- Examples:

OPERAND1	CONDITIONAL OPERATOR	OPERAND2
P_PRICE	>	10.00
V_STATE	=	FL
V_CONTACT	LIKE	Smith%
P_QOH	>	P_MIN * 1.10

- An operand can be:
  - A simple column name such as P_PRICE or V_STATE.
  - A literal or a constant such as the value 10.00 or the text 'FL'.
  - An expression such as P_MIN * 1.10.

KNOWLEDGE & SOFTWARE ENGINEERING

Conditional Expressions

Some **common practices** used to write efficient conditional expressions:

- *Use simple columns or literals as operands in a conditional expression—avoid the use of conditional expressions with functions whenever possible.*

  Examples:

  - P_PRICE > 10.00 is faster than P_QOH > P_MIN * 1.10
    - because the DBMS must evaluate the P_MIN * 1.10 expression first.
  - UPPER (V_NAME) = 'JIM' is slower V_NAME = 'Jim' if all names in the V_NAME column are stored with proper capitalization

KNOWLEDGE & SOFTWARE ENGINEERING

Conditional Expressions

- *Numeric field comparisons are faster than character, date, and NULL comparisons.*
  - CPU handles numeric comparisons (integer and decimal) faster than character and date comparisons
  - Indexes do not store references to null values
    - involve additional processing
    - Tend to be the **slowest** of all conditional operands.

KNOWLEDGE & SOFTWARE ENGINEERING

Conditional Expressions

- *Equality comparisons are faster than inequality comparisons*
  - Example:
    - P_PRICE = 10.00 is processed faster → a direct search using the index in the column.
    - An inequality symbol (>, >=, <, <=) → There will almost always be more “greater than” or “less than” values than exactly “equal” values in the index → need additional processing to complete the request
  - The slowest of all comparison operators is LIKE with wildcard symbols, as in V_CONTACT LIKE “%glo%”
    - With the exception of null (see previous slide)
  - “not equal” symbol (<>) yields slower searches, especially when the sparsity of the data is high

KNOWLEDGE & SOFTWARE ENGINEERING

Conditional Expressions

- *Whenever possible, transform conditional expressions to use literals*
  - Examples:
    - change P_PRICE - 10 = 7 to P_PRICE = 17
- *When using multiple conditional expressions, write the equality conditions first.*
  - Remember: *Equality comparisons are faster than inequality comparisons*
  - Examples:
    - change P_QOH < P_MIN AND P_MIN = P_REORDER AND P_QOH = 10 to P_QOH = 10 AND P_MIN = P_REORDER AND P_MIN > 10

KNOWLEDGE & SOFTWARE ENGINEERING

Conditional Expressions

- *If you use multiple AND conditions, write the condition most likely to be false first*
  - DBMS will stop evaluating the rest of the conditions as soon as it finds a conditional expression that is evaluated as false.
  - Example:
    - P_PRICE > 10 AND V_STATE = ‘FL
    - If you know that only a few vendors are located in Florida, you could rewrite the condition as: V_STATE = 'FL' AND P_PRICE > 10
- *When using multiple OR conditions, put the condition most likely to be true first.*
  - DBMS will stop evaluating the remaining conditions as soon as it finds a conditional expression that is evaluated as true.

KNOWLEDGE & SOFTWARE ENGINEERING

Conditional Expressions

- *Whenever possible, try to avoid the use of the NOT logical operator.*
  - Example:
    - NOT (P_PRICE > 10.00) can be written as P_PRICE <= 10.00
    - NOT (EMP_SEX = 'M') can be written as EMP_SEX = 'F'

KNOWLEDGE & SOFTWARE ENGINEERING

Query Formulation

- To formulate a query, you would normally follow the steps outlined below:
  - *Identify what columns and computations are required*:
    1. What columns do you need? All columns, or only certain columns?
       - If you need only certain columns don't use select *, since the size of data to be transferred will be greater
    2. Do you need simple expressions? Do you need functions? Do you need aggregate functions?
    3. Determine the granularity of the raw data required for your output.
       - You might need to summarize data not readily available on any table
       - You might consider breaking the query into multiple subqueries and storing those subqueries as views

KNOWLEDGE & SOFTWARE ENGINEERING

Query Formulation

- *Identify the source tables*
  - Try to use the least number of tables in your query to minimize the number of join operations
- *Determine how to join the tables*
  - Properly identify what and how to join the tables
  - In some cases, you will need some type of natural join, but others, you might need to use an outer join

KNOWLEDGE & SOFTWARE ENGINEERING

Query Formulation

- Determine what selection criteria is needed.
  - Simple comparison. For example, P_PRICE > 10.
  - Single value to multiple values. For example, V_STATE IN ('FL', 'TN', 'GA').
  - Nested comparisons → nested selection criteria involving subqueries. For example: P_PRICE >= (SELECT AVG(P_PRICE) FROM PRODUCT).
  - Grouped data selection → the HAVING clause
- *Determine in what order to display the output.*
  - you may need to use the ORDER BY clause → ORDER BY clause is one of the most resource-intensive operations for the DBMS.

KNOWLEDGE & SOFTWARE ENGINEERING