T-SQL Performance Recommendations

SQLBits9 - Liverpool, 1st October 2011

About Me

Dipl.-Ing. Milos Radivojevic, Vienna, Austria

Mentor with SolidQ

10+ years SQL Server experience: MCTS, MCP, MCT
-Relational and DWH/BI area
Workshop: SQL Server for Application Developers
Conference Speaker: SQLBits, SQLU Summit, SQL UG Austria

Microsoft CERTIFIED Technology Specialist
SQL Server 2005
SQL Server 2008, Database Development

sqlbits

SQLUSUMMIT September 21-23 2011 Budapest

Contact:
mradivojevic@solidq.com
www.solidq.com

About SolidQ

SolidQ provides the world's most trusted and reliable business intelligence, data management, collaboration and custom solutions for Microsoft's cloud and on-premise platforms.

- 80 of the world's top information experts in our BI and data management team, including more than 30 SQL MVPs
- Authored over 30 industry leading books
- Frequent speakers at local and international SQL Server technical conferences (TechEd, SQL PASS Summit, SQL Saturdays…)
- Serving over 800 clients in 22 countries

www.solidq.com

sqlbits

SolidQ™ Journal

- Free, monthly e-magazine providing answers you need to be successful with your Microsoft SQL Server, Business Intelligence, and software development solutions
- Featuring in-depth technical articles, real-life case studies, practical columns, and seasoned perspectives from our lineup of influential global experts
- Both inside SolidQ and around the community

www.solidq.com/sqj

Agenda

- General Recommendation
- Functions in the WHERE Clause
- Data Type Conversions in WHERE Clause
- SARGable and Non-SARGable
- Local Variables
- Database Constraints and Performance
- UDFs: Scalar vs. Inline Table
- Other Recommendations

General Recommendation

- The query optimizer does a good job
- It considers a lot of possible plans and chooses a very good one
- Almost always this is an optimal execution plan

- **Recommendation**: Do not write a code which limits optimization options for the query optimizer!

Functions in the WHERE Clause

- A function call on a WHERE clause column prevents the optimizer from using an appropriate index operator
- Function is evaluated for each row => the optimizer uses in this case TableScan or IndexScan operator instead of appropriate Seek operator

- **Recommendation:** Function should be called against parameters, table columns should not be used as arguments!

Data Type Conversions

- Implicit conversion
- Estimation problem with LIKE operator

- **Recommendation:** Avoid string conversions especially when involved columns are “varchar” or “char” data type

Non-SARGable WHERE Clause

- Search ARGument ABLE - sargeable predicate is one in which an index can be used
- Sargable: =,>,<,>=,<=, BETWEEN, LIKE without leading asterisk
- Non-sargable <>, IN, OR, NOT IN, NOT EXISTS, NOT LIKE, LIKE with leading %

- **Recommendation:** Avoid non-sargable predicates and replace them with sargable equivalents, when it's possible

Local Variables

- By using a local variable the optimizer cannot generate an optimal execution plan because the variable value is not known at the compile time

- **Recommendation:** Understand the behavior of local variables and how they affect an execution plan

Database Constraints and Performance

- Foreign Keys
- Unique Constraints
- Check Constraints

- **Recommendation:** Use database constraints – they are not important for consistence only but also for performance!

Other Recommendations

- VERIFY EXISTENCE - EXISTS vs. COUNT(*)
  - Data existence verifying performs same for both versions
- UNION vs. UNION ALL
  - Use UNION ALL when you know that sets are not overlapped and when duplicates are allowed
- IN vs. EXISTS
  - From version 2005 perform same
- The SQL Server optimizer should have all relevant information and only then it can generate an optimal execution plan.

Other Recommendations

- SELECT ONLY REQUIRED COLUMNS
  - By using a local variable the optimizer cannot generate an optimal execution plan because the variable value is unknown at compile time
- USE ORDER BY ONLY WHEN IT'S NECESSARY
  - Data existence verifying perform same, but when local variables the whole table will be scanned
- Only required information should be requested. It sounds trivial, but there are lot of examples with unnecessary SELECT * or ORDER BY statements

Other Recommendations

- Use SET NOCOUNT ON - After every query in batch or SP is executed, the server reports the number of rows affected => network overhead
- Reduce lock overhead by using NOLOCK hints
  - Be careful regarding inconsistency problems, but be pragmatic, there is a lot of cases where these problems cannot occur or it doesn't matter

UDFs: Scalar vs. Inline Table

- Scalar UDF is called for each row
- Inline TVF will be expanded and an optimal plan will be generated
- **Recommendation:** Scalar UDF is more intuitive, but an inline function performs better!

Next Sessions

Speaker	Title	Room
Ola Hallengren	Inside Ola Hallengrens Maintenance Solution	Aintree
Mark Pryce-Maher	Building a SSMS Add-in; The Agony and Ecstasy	Lancaster
Marco Russo	Vertipaq vs OLAP: Change Your Data Modeling Approach	Pearce
Gert Drapers	Database Development with SQL Server Juneau	Boardroom
Satya Jayanty	SQL Server Upgrade: take help from tools and best practices	Empire
Emil Glownia	Advanced SSRS. Find out what you have been missing.	Derby

twitter #SQLBITS

- Contact:

  mradivojevic@solidq.com

  www.solidq.com

Thank You!