mayato

Was wir vorhersagen, soll auch eintreffen!

Optimale Performance: SAS® und EXASOL - Datenbankschnittstelle richtig nutzen //

Benno Schultheiss, Annkathrin Wagner, Paolo Leon Vacilotto

Zusammenfassung

In diesem Whitepaper werden grundlegende Regeln bei der Verwendung von SAS® in Verbindung mit der EXASolution Datenbank des Hersteller EXASOL aufgezeigt und demonstriert, welche Performance-verbesserungen für ETL-Prozesse bei der Optimierung von SAS®-Programmierungen erzielt werden können.

Schlüsselwörter: SAS®, EXASOL, EXASolution, Datenbank, PROC SQL, Performance, ETL

Einleitung

Das Volumen und die Komplexität der Daten in modernen Unternehmen stellen eine große Herausforderung für IT-Systeme dar. Gleichzeitig werden die Ansprüche an die Performance und Zuverlässigkeit der Prozesse innerhalb einer IT-Landschaft immer höher. Konventionelle IT-Architekturen und klassische Technologien können mit den Anforderungen von Real-Time-Analyse und Self-Service Business Intelligence Lösungen nicht mithalten. Insbesondere Analysetools sind dabei auf eine schnelle Datenbereitstellung aus dem Data Warehouse angewiesen. Die Anwender erwarten höchste Performanz bei der Verarbeitung und der Auswertung ihrer Daten, leicht bedienbare Analyseverfahren sowie moderne Frontends zur interaktiven Visualisierung der Zusammenhänge. Bis heute hat es kein Softwarehersteller geschafft, Werkzeuge auf höchstem Niveau in all diesen Bereichen anzubieten.

Eine optimale Basis für analytische Fragestellungen bietet die für Analytics optimierte Datenbank des unangefochtenen Technologieführers EXASOL mit Sitz in Nürnberg. EXASOL ist auf die Bereitstellung einer hoch performanten Datenbank spezialisiert. Die hohe Performanz von Datenverarbeitung und Abfragen wird grundsätzlich durch In-Memory-Technologie und ein hohes Maß an Parallelisierbarkeit auch auf sehr großen Mehrknotenarchitekturen erzielt. Vor allem das außergewöhnlich intelligente Verfahren zur optimalen Nutzung des Arbeitsspeichers, d.h. die Entscheidung, welche Daten wann auf welchem Rechner in Memory gehalten werden, erlaubt EXASOL einen entscheidenden Wettbewerbsvorteil. So ist EXASolution nicht nur ultraschnell, sondern es ist dies auch praktisch ohne manuelle Eingriffe und entsprechend wartungsarm.

Die Datenbank kann mit einfachen Mitteln in Analyse-Software wie z. B. SAS® Enterprise Guide integriert werden. SAS® kommuniziert mit relationalen Datenbanken via Treiber (klassischer Weise ODBC) unter Verwendung von Data Step Statements und SQL. Um das Potential der Hochleistungsdatenbank jedoch vollständig zu nutzen, ist eine Anpassung der Abfragen auf die Eigenschaften der EXASOL Datenbank ratsam. In diesem Whitepaper werden grundlegende Regeln zur Verwendung von SAS® auf EXASOL Basis aufgezeigt und demonstriert, wodurch Performanceverbesserungen bei der Optimierung von SAS®-Programmierungen für ETL-Prozesse erzielt werden können.

Test-System

Die virtualisierte Test-Systemlandschaft besteht aus drei virtuellen Servern:

- SAS® Terminal Server (Windows)
- SAS® 9.4 Server (Red Hat Linux)
- EXASOL Datenbank Server (Red Hat Linux)

Die Systeme laufen in einer Virtual Private Cloud und sind somit durch eigene Subnetze und Firewalls von den restlichen Clouds bzw. vom Internet getrennt. Der Zugriff erfolgt ausschließlich über den Terminal Server, welcher über eine Remote-Desktop-Verbindung gesteuert wird.

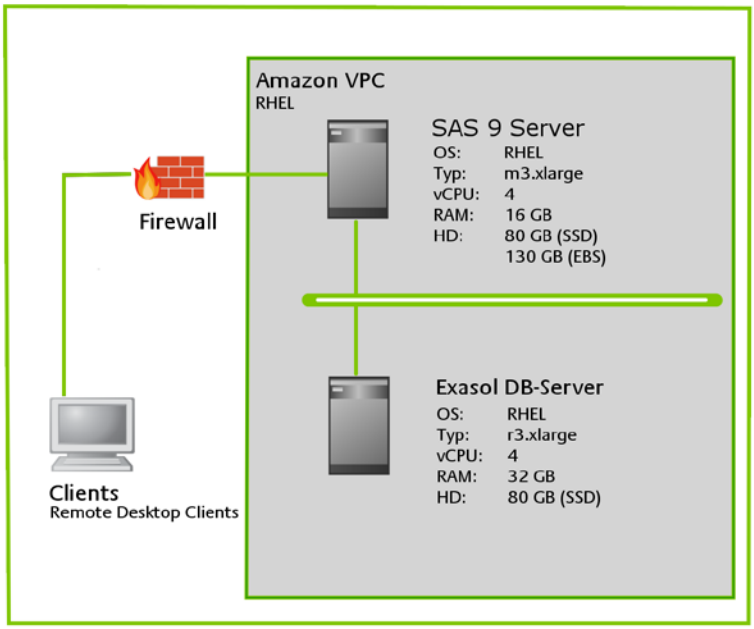

Abbildung 1: Server-Topologie

Datengrundlage

Die Verbindung zwischen einer EXASolution Datenbank und SAS®-Enterprise Guide wurde anhand des NYC-Flights Datensatzes getestet. Der Datensatz ist im Netz als CSV-Datei frei verfügbar und bietet die Möglichkeit, einzelne Tabellen des Datensatzes (Flights, Airports, Carriers) miteinander zu fusionieren.

Datengrundlage			
Datensatz	**Größe**	**Zeilen**	**Spalten**
Flights	3.331.055 KB	35.850.733	29
Airports	239 KB	3.376	7
Carriers	43 KB	1.491	2

Tabelle 1: Übersicht der Datensätze

Operationen und Einstellungen

Im folgenden Abschnitt werden grundlegende Operationen demonstriert und aufgezeigt, wie mit zusätzlichen Einstellungen in SAS® Verbesserungen der Performance zwischen SAS® und EXASolution möglich ist.

1. Beladen der Datenbank

Unter der Prämisse, dass die Testdaten erst in die EXASolution geladen werden müssen, wird zunächst gezeigt, wie mit dem Data Integration Studio von SAS® der Ladevorgang konfiguriert wird.

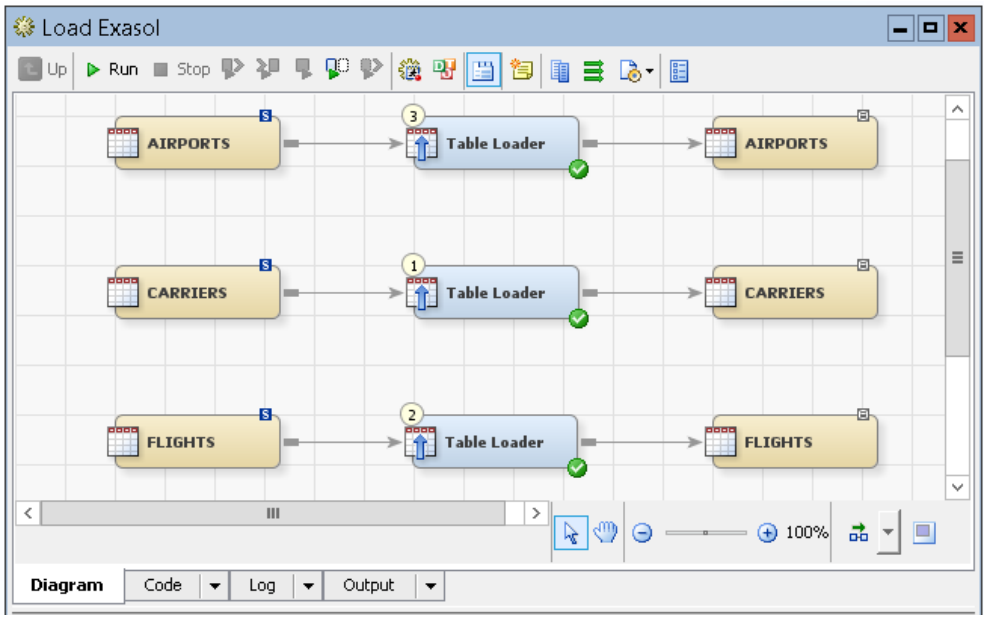

Abbildung 2: SAS® Data Integration Studio - Beladen der EXASolution mit Table Loader

Hierbei wird der Table-Loader verwendet, um die Spalten der Quelldatei mittels Mapping der Zieltabelle auf der Datenbank zuzuweisen.

Wird dazu das SAS®-Skript betrachtet, zeigt sich, wie durch den Table-Loader die Einstellungen der Quellordner festgelegt werden und die Verbindung zur Datenbank aufgebaut wird:

LIBNAME legt den Namen einer Library fest und bezieht sich im hier vorliegenden Fall auf einen lokalen Pfad als Quelle für die Dateien.

```
LIBNAME M_DM16 BASE "/SAS/SASData/mayato/data/DataMiningStudie2016";
```

Ein zweites LIBNAME Statement benennt die Library M_EXAS und öffnet die Verbindung zum Datenbank Server mit entsprechenden Login-Daten.

```
LIBNAME M_EXAS ODBC DATASRC=mrhelexasol SCHEMA=MAYATO  USER="USERNAME"
PASSWORD="USERPW";
```

Bei größeren Dateien kann bereits hier festgestellt werden, dass eine Anpassung der Standardeinstellungen sinnvoll ist, da der Ladevorgang sehr lange dauern kann.

2. Optimierung des Ladevorgangs – INSERTBUFF / READBUFF

Einer der schnellsten Wege, große Datensätze in eine relationale Datenbank zu laden, ist die Verwendung der Bulk-Load Fähigkeit, welche in vielen Datenbanken integriert ist. EXASolution bietet hierfür einen internen Service (den sog. EXAloader) mit schnellen, parallelen Ladekapazitäten. EXAloader kümmert sich um den Verbindungsaufbau zu SAS® und verteilt die Daten optimal im EXASolution-Cluster. Im Falle, dass eine Datenübertragung explizit von SAS® ausgeführt werden soll, existieren einige SAS®-Optionen zur Verbesserung der Performance.

LIBNAME- und Dataset-Optionen legen fest, wie viele Zeilen von einer Datenbanktabelle gelesen oder beschrieben werden können. Meist ist hier in SAS® eine niedrige Anzahl (1) als Standardwert festgelegt.

Bei Veränderung des READBUFF und INSERTBUFF in der LIBNAME Option (z.B. auf 5.000) kann eine erhebliche Verbesserung der Performance erzielt werden.

Als Beispiel für die Einstellung einer LIBNAME-Option dient folgender Code:

```
LIBNAME EXA_1 ODBC DATASRC=mrhelexasol SCHEMA=MAYATO
AUTHDOMAIN="EXASOLAuth" insertbuff=5000 dbcommit=5000;
```

DBCOMMIT legt fest, nach wie vielen verarbeiteten Zeilen ein Schreibvorgang auf der Datenbank erfolgt.

Die nachstehende Grafik zeigt die Auswirkung der INSERTBUFF-Option gegenüber den Standard-Einstellungen von SAS®. In diesem Test wurde eine neue Tabelle auf der EXASOL-Datenbank mittels PROC SQL Prozedur in SAS® Enterprise Guide erstellt.

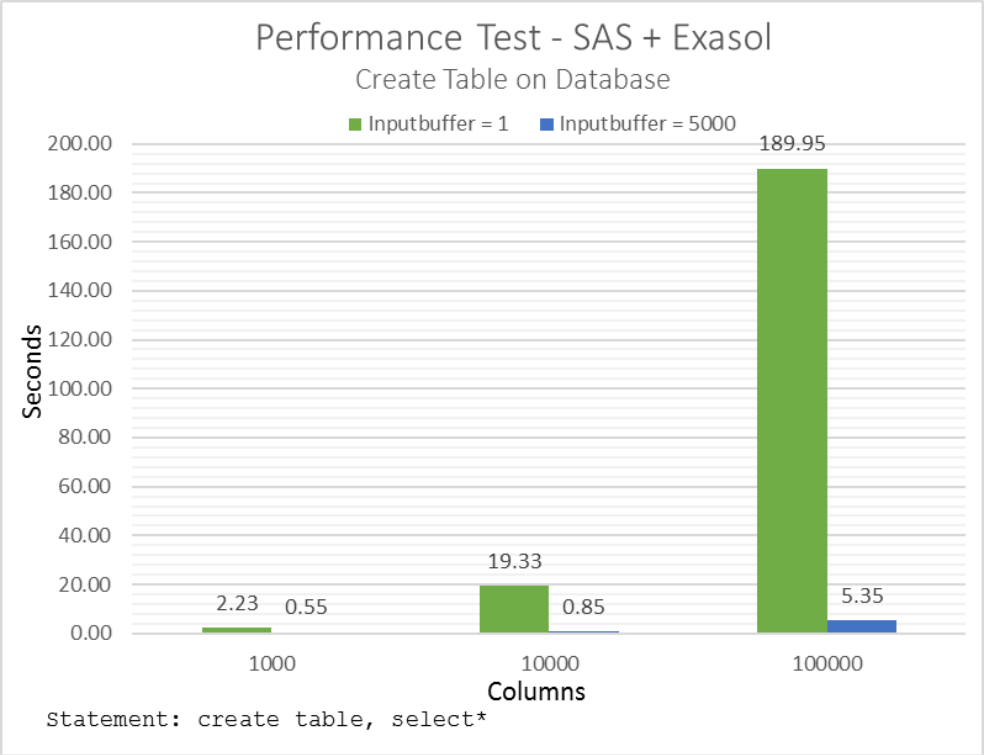

Abbildung 3: Performance-Test Ergebnisse - Veränderung der LIBNAME Optionen

Der Test zeigt auf, dass die Einstellung des Readbuffers sich mit steigender Datenmenge immer stärker auf Abfragen von SAS® an die EXASOL-Datenbank - durch die effizientere Übertragung der Daten von EXASolution an SAS® - auswirkt.

Der optimale Wert für diese LIBNAME-Optionen wird durch Faktoren wie Netzwerktyp und verfügbarem Speicher beeinflusst und kann individuell festgestellt werden.

3. Simple Select Query

Nach erfolgreicher Beladung der Daten können mit Hilfe der PROC SQL Prozedur gezielte Abfragen erstellt werden. Folgendes Code-Beispiel zeigt ein simples SELECT-Statement mit Nutzung des WHERE-Clause, um nur Datenzeilen mit einem bestimmten Datum zu erhalten.

```
proc sql;
      create table work.test as
      select * from exa_1.flights
      where flightdate = '03Jan08'd;
quit;
```

4. Pass-Through - Unterschiede zwischen implicit- und explicit-Pass-Through

In vielen Fällen wird über simple SELECT-Statements hinaus eine Einschränkung der Datenauswahl oder unterschiedliche Funktionen zur Datentransformation verwendet. Das Verwenden von SAS®-Funktionen oder Datenbankfunktionen kann einige Tücken aufweisen und die Performance erheblich beeinträchtigen.

Die Pass-Through-Option ermöglicht eine direkte Verbindung zur relationalen Datenbank (RDBMS). Innerhalb dieser Verbindung können sowohl Query- als auch non-Query SQL-Statements an die Datenbank zur Verarbeitung gesendet werden. Die SQL Pass-Through Facility führt dabei folgende Schritte durch:

1. Aufbau einer Verbindung zur Datenbank mittels CONNECT-Statement
2. Senden einer non-Query, RDBMS-spezifischen SQL-Statements mittels EXECUTE-Statement

Der Unterschied zwischen implicit- und explicit-Pass-Through kann für die Performance enorm hoch ausfallen, denn bei implicit-Pass-Through wird SAS®-SQL verwendet und entsprechend SAS® den Aufbau der Verbindung zur Datenbank und die Übersetzung des Codes in die jeweilige Sprache der Datenbank überlassen. Je nach Datenbank kann hierbei der gesamte Befehl oder nur Teile des Befehls übersetzt werden. Somit kann es vorkommen, dass Teile der Verarbeitung des SQL-Statements von SAS® übernommen werden. Wenn Daten aus der Work-Library und dem Datenbank-Server verwendet werden, zieht SAS® die Daten vom SQL-Server und verarbeitet diese intern. Dies kann zu längerer Prozessdauer führen, da SAS® in diesen Fällen nicht von der performanten in-Memory Technologie der EXASOL Datenbank profitieren kann.

Untenstehende Code-Beispiele verdeutlichen den Unterschied zwischen implicit- und explicit-Pass-Through.

Die erste implicit-Abfrage (Implicit 1) wird in SAS®-SQL-Code verfasst und bei Ausführung von SAS® in die Datenbank-Sprache übersetzt. In EXASolution existiert jedoch kein äquivalentes Statement für die Funktion „intnx()“. Es führt zu Performanceverlust, wie sich später im Performance-Test zeigt. Der Code verursacht, dass jeder Wert der Variable „flightsdate“ aus der Datenbank nach SAS® gezogen und mittels „`intnx()`“-Funktion gegenüber dem Datum `'01Jan08'd` vergleichen wird.

Implicit 1:

```
proc sql;
      create table work.test as
      select * from exa_1.flights
      where intnx('day', flightdate, -3) = '01Jan08'd;
quit;
```

In folgendem Codebeispiel wurde die „`intnx()`“-Funktion so eingesetzt, dass der resultierende Datumswert direkt auf der EXASOL-Datenbank mit der Variable „flightsdate“ abgeglichen werden kann. Dies vermeidet unnötigen Datentransfer zwischen SAS® und EXASolution.

Implicit 2:

```
proc sql;
      create table work.test as
      select * from exa_1.flights
```

```
        where flightdate = intnx('day', '01Jan08'd, +3);
quit;
```

Bei einem explicit SQL Pass-Through wird die Abfrage über die Verbindung zur Datenbank direkt auf der Datenbank ausgeführt. Somit lassen sich Befehle, die von SAS® nicht in die Sprache der ODBC übersetzt werden können, durchführen. Diese müssen jedoch der jeweiligen Datenbanksyntax entsprechen.

Explicit:

```
proc sql;
        connect to ODBC (DATASRC=mrhelexasol AUTHDOMAIN="EXASOLAuth");
        create table work.test as
        select * from connection to ODBC
        (
        select * from mayato.flights
        where (flightdate - interval '3' day) = '2008-01-01'
        );
        disconnect from ODBC;
quit;
```

Die Code-Beispiele weisen im durchgeführten Performance-Test mit dem Flights-Datensatz erhebliche Unterschiede in der Prozessdauer auf. Es wird aufgezeigt, wie Prozesse Performance verlieren, wenn SAS® die intnx()-Funktion nicht direkt in die EXASOL-Sprache übersetzen kann. Wird allerdings die Funktion für EXASolution umgeschrieben oder ein funktional gleiches Statement mit Pass-Through direkt auf der Datenbank ausgeführt, verringert sich die Laufzeit erheblich. Dies ist im Testergebnis in Abbildung 4 ersichtlich.

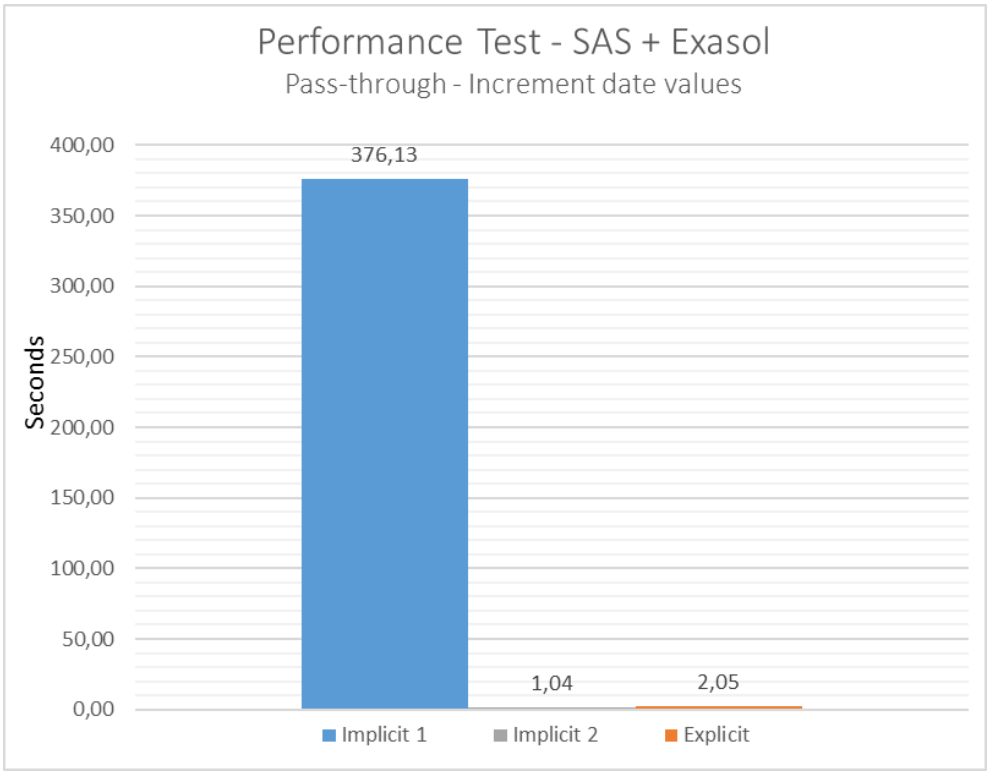

Abbildung 4: Performance Test Ergebnisse – PROC SQL mit implicit und explicit Pass-Through

5. Joins – Aufbau und Optimierung

Joins sind häufig verwendete und ressourcenintensive Operationen des ETL-Prozesses. In diesem Abschnitt wird gezeigt, wie bei Joins die Performance durch WHERE-Statements beeinflusst wird. Wie im vorherigen Abschnitt gezeigt wurde, kann die Art des Statements als implicit oder explicit Pass-Through starke Auswirkungen haben. Dies ist auch bei Join-Statements der Fall.

Bei einem homogenem Join auf der Datenbank werden die Algorithmen der Datenbank eingesetzt. Folgende Test-Ergebnisse demonstrieren Best-Practices, welche die Performance von Join-Algorithmen verbessern und zur optimalen Nutzung der EXASOL In-Memory-Technologie verhelfen.

Abbildung 5 zeigt den Aufbau von Inner-Joins mit zwei Tabellen aus der EXASOL-Datenbank in SAS® Data Integration Studio. Innerhalb der Join-Konfiguration wird das WHERE-Statement, neben dem Abgleich für den Join, auch für eine Einschränkung der Auswahl mittels einer Datum-Variable genutzt. In diesem Beispiel wird die Ergebnistabelle eines Joins jeweils direkt auf der EXASOL-Datenbank gespeichert.

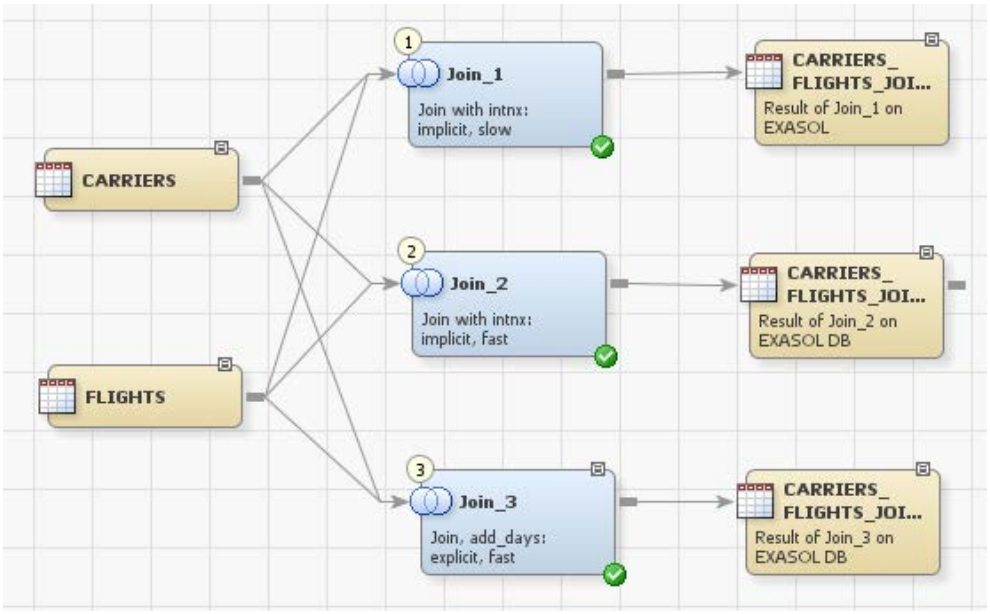

Abbildung 5: Job-Aufbau, Table-Joins on EXASolution

Die Performance der einzelnen Joins unterscheidet sich stark. Insbesondere implicit-Statements können bei suboptimalem Einsatz von SAS®-Funktionen wie bspw. „intnx()“ zu sehr langen Prozessen führen. Korrekt eingesetzte explicit-Statements zeigen sehr gute Performance, wie sich am Beispiel der „add_days“-Funktion von EXASOL zeigt. Diese Funktion erfüllt im verwendeten WHERE-Statement die gleiche Funktion wie die SAS®-Funktion „intnx()“.

Folgende Grafik zeigt die Ergebnisse der Performance-Tests. Die farbliche Unterteilung identifiziert den physikalischen Speicherort der Ergebnistabelle (grün – EXASOL Datenbank bzw. blau SAS® lokal).

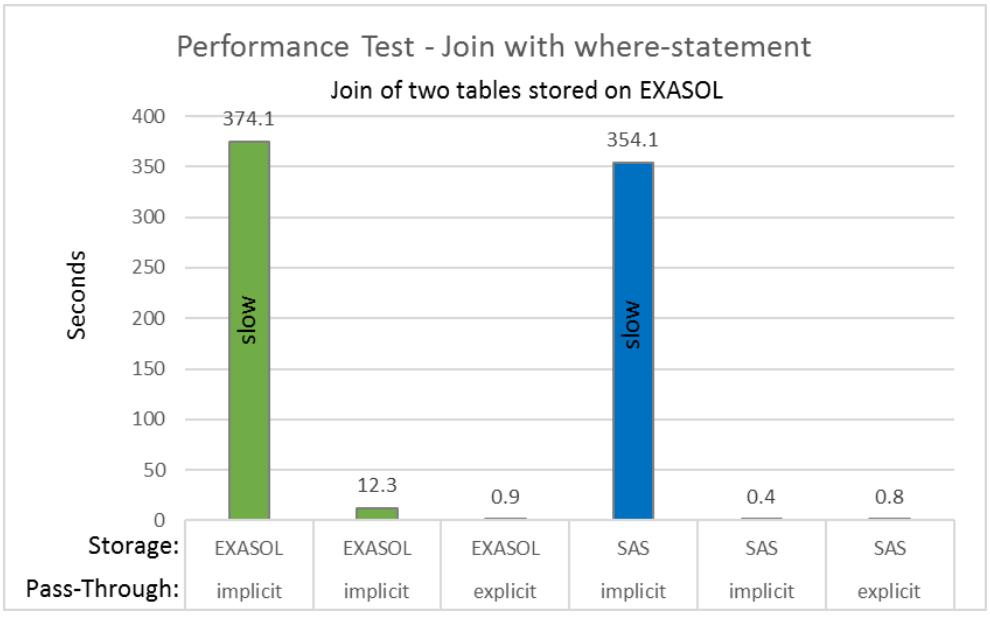

Abbildung 6: Performance Testergebnisse - Table Joins, Data on EXASolution

Die Verbesserung der Performance wurde durch Anpassungen des WHERE-Statements erreicht. Der interne Aufbau der Join-Funktion zeigt die Platzierung der WHERE-Funktion (siehe Abbildung 7). Wird in diesem WHERE-Statement beispielsweise die intnx()-Funktion für eine Selektion nach einem bestimmten Datumswert verwendet, ohne dabei den Prozessaufwand zu optimieren, ergibt sich eine unnötig lange Laufzeit, wie in der Auswertung dargestellt (374,1 Sekunden).

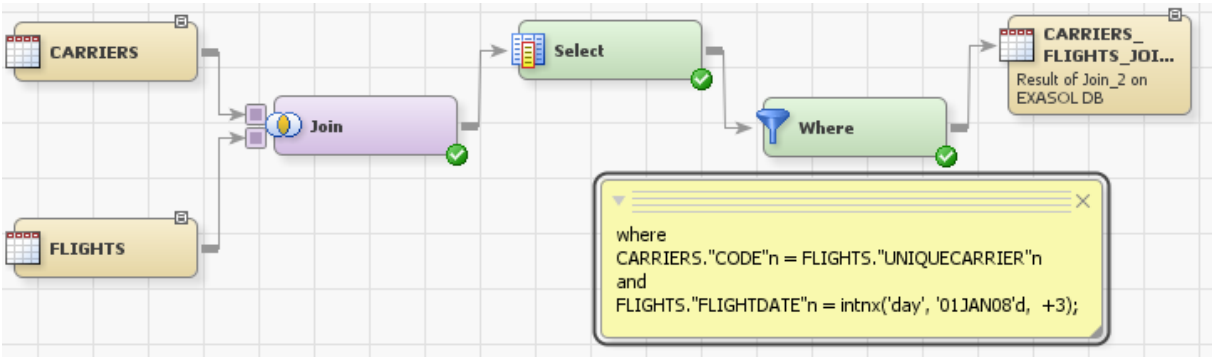

Abbildung 7: Aufbau Inner-Join mit WHERE-Funktion

Ein leichte Anpassung des WHERE-Statements in SAS® - wie in der gelben Annotation von Abbildung 7 gezeigt - führt auch in Joins zu einer Verringerung der Laufzeit. Allerdings ist diese, mittels implicit-Pass-Through übermittelte Ausführung, mit 12,3 Sekunden Prozesslaufzeit immer noch nicht optimal.

Eine erneute Anpassung des WHERE-Statements und Einstellung als explicit-Pass-Through führt zu einer weiteren Verbesserung, welche eine Prozesslaufzeit von 0,9 Sekunden erreicht:

```
where
        CARRIERS."CODE" = FLIGHTS.UNIQUECARRIER
        and FLIGHTS.FLIGHTDATE = ADD_DAYS(DATE '2008-01-01', 3);
```

Die im obigen Code-Ausschnitt verwendete “ADD_DAYS”-Funktion stellt eine alternative Methode zu der in Abschnitt 4 verwendeten „interval“-Funktion des Explicit-Code-Beispiels dar. In beiden Fällen werden mit EXASOL-Funktionen drei Tage zu einem vorgegebenen Datum hinzugefügt, welches dann mit den Werten der FLIGHTDATE-Variable verglichen wird.

In einem weiteren Test wurde die Zieltabelle der Joins auf eine lokale, temporäre SAS®-Library geändert. Bei sonst gleichem Aufbau der Joins, wie im zuvor beschriebenen Test, zeigen Anpassungen des WHERE-Statements bei implicit-Pass-Through die beste Performance. Die in blau dargestellten Balken von Abbildung 6 zeigen, dass das nicht optimierte WHERE-Statement 354.1 Sekunden und die optimierte Variante 0.4 Sekunden Prozesslaufzeit ergeben. Damit ist letztere Variante sogar performanter als das angepasste explicit-Pass-Through Statement (0.8 Sekunden Laufzeit).

Im letzten beschriebenem Join-Test lagen die Tabellen des Joins gemeinsam auf der EXASolution. In einem weiteren Test wurde untersucht, welche Auswirkungen eine Verteilung der beiden Tabellen auf SAS® und EXASolution bei einem Join verursacht (heterogener Join). Vorweg ist festzuhalten, dass ein explicit-Pass-Through bei einem heterogenem Join nicht möglich ist.

Abbildung 8 zeigt den Aufbau des Inner-Joins mit CARRIERS-Tabelle aus einer lokalen SAS®-Library und FLIGHTS aus der EXASOL-Datenbank. Die resultierende Tabelle wird auf EXASolution gespeichert. Der Join ist mit einer Prozessdauer von 15,1 Sekunden schlecht performant.

Abbildung 8: Aufbau, Inner-Join mit Tabellen aus SAS® und EXASolution, implicit Pass-Through

Bei heterogenen Join-Vorgängen kann es besonders dann zu Performanceproblemen kommen, wenn SAS® SQL einen Hash Join anwendet. SAS® SQL versucht zunächst, das WHERE-Statement auf der Datenbank auszuführen, zieht anschießend die Daten von der Datenbank nach SAS® und führt dort den Join mit den lokalen SAS® Daten aus.

Standard heterogene Jonis können vor allem bei großen Datensätzen, die zunächst in SAS® eingelesen werden müssen, zu Performanceproblemen führen. Bei leistungskritischen Vorgängen stehen drei Möglichkeiten zur Verfügung:

- Kopieren der SAS®-Daten in die Datenbank und Join auf der Datenbank -> hierfür können auch temporäre Tabellen in der Datenbank angelegt werden, in den meisten Situationen die beste Variante.
- Kopieren der Datenbank Daten in SAS® Data-Sets und Join in SAS® -> damit werden Operationen nur in SAS® durchgeführt.
- Erhöhung der Join-Performance in SAS® durch Sortierung anhand von Join-Keys -> funktioniert nur, wenn Daten in SAS® vorliegen.

Das Kopieren der Daten zwischen SAS® und der Datenbank kann großen Performanceverlust generieren. Ergo empfiehlt es sich, die Daten auf die EXASOL-Datenbank zu transferieren, da dann die hohe Performance der EXASolution den Join-Vorgang übernimmt.

Der Performance-Test in Abbildung 9 zeigt, dass auch bei dieser Vorgehensweise ein erheblicher Leistungsunterschied zwischen explicit und implicit-Pass-Through entsteht.

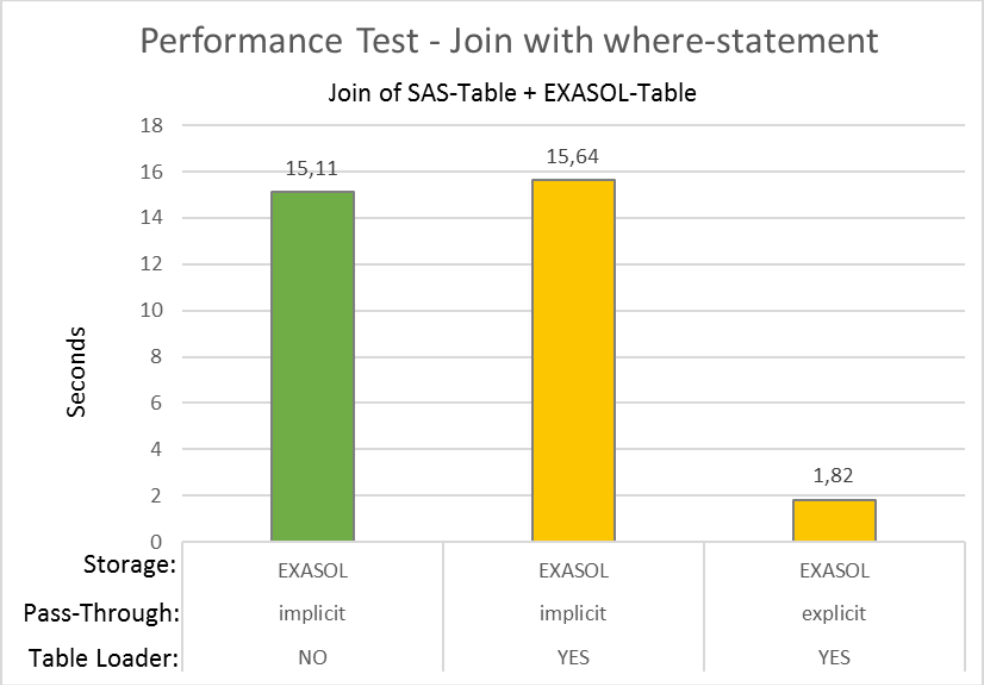

Abbildung 9: Performance Testergebnisse – Joins von Tabellen aus SAS® und EXASolution

Die orange gefärbten Balken zeigen an, dass vor dem Join zunächst ein Transfer der CARRIERS-Tabelle zur EXASolution mittels der Table-Loader-Funktion vorgenommen wurde. Die explicit-Pass-Through-Variante stellt in diesem Fall die schnellste Möglichkeit zum Join dar.

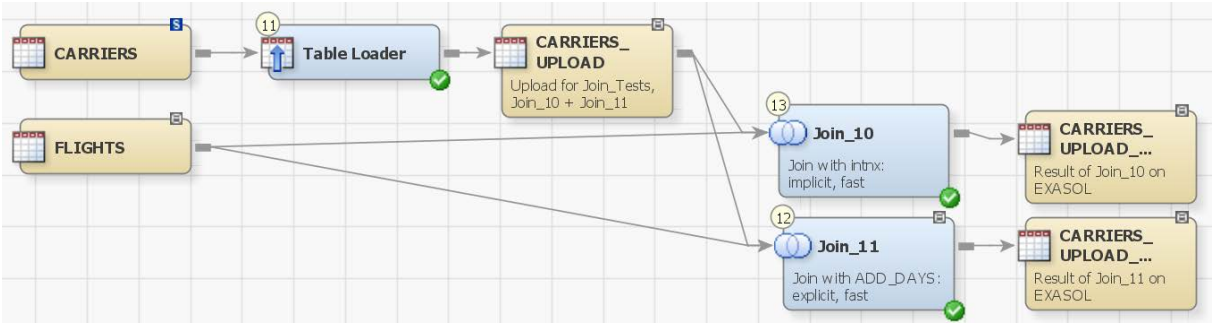

Abbildung 10: Aufbau, Table-Loader und Inner-Joins mit Tabellen aus SAS® und EXASolution

Abbildung 10 zeigt den Aufbau des Jobs. In beiden eingebauten Varianten (implicit und explicit-Pass-Through) wird die vom Table-Loader auf EXASolution transferierte Tabelle für den Join verwendet. Damit ist der Join ein homogener Join und ein explicit-Pass-Through ist wieder möglich.

Anzumerken ist, dass die verhältnismäßig geringe Größe der CARRIERS-Tabelle obiges Vorgehen, die Tabelle vor dem Join nach EXASolution zu laden, begünstigt. Im Falle, dass die deutlich größere FLIGHTS-Tabelle nur lokal in SAS® vorliegt, wäre ein Transfer für einen einmaligen Join kaum rentabel. Die Entscheidung, welche Vorgehensweise zur Performance-Optimierung genutzt wird, ist also immer auch unter den Gesichtspunkten der Größe der Tabellen, der Anzahl der Tabellen und der Häufigkeit des Einsatzes mit performance-relevanten Funktionen, zu treffen.

6. SASTRACE – Was passiert auf der Datenbank?

Um Abfragen auf der EXASOL-Datenbank zu optimieren, ist es wichtig, einen Einblick darin zu erhalten, welche SQL-Befehle von SAS® an die Datenbank gesendet werden. Die SASTRACE und SASTRACELOC-Optionen helfen hierbei, in dem sie offenlegen, welche SQL-Befehle auf EXASolution ausgeführt wurden.

```
OPTIONS SASTRACE=',,,d'SASTRACELOC=saslog NOSTSUFFIX;
```

Zeigt das Logfile nach dem Ausführen einer Abfrage, dass ein Teil der Query nicht auf der Datenbank ausgeführt werden konnte, wird dieser Teil stattdessen in SAS® ausgeführt. In den meisten Fällen sollte dies vermieden werden, denn Ziel ist es, die EXASolution so viel Arbeit wie möglich übernehmen zu lassen, um von den Performance-Vorteilen zu profitieren.

Dieser Ausschnitt aus dem Logfile eines simplen SELECT-Statements zeigt, dass die EXASOL-Datenbank die Ausführung vollständig übernommen hat:

```
          proc sql;
25           create table exa_1.sashelp_cars_2 as
26           select * from sashelp.cars;
  27 1481557286 no_name 0 SQL (2)
ODBC_3: Prepared: on connection 1 28 1481557286 no_name 0 SQL (2)
SELECT * FROM MAYATO.SASHELP_CARS_2 WHERE 0=1  29 1481557286 no_name 0 SQL (2)
  30 1481557286 no_name 0 SQL (2)
ODBC: AUTOCOMMIT is NO for connection 2 31 1481557286 no_name 0 SQL (2)
NOTE: SAS variable labels, formats, and lengths are not written to DBMS tables.
  32 1481557286 no_name 0 SQL (2)
ODBC_4: Executed: on connection 2 33 1481557286 no_name 0 SQL (2)
```

Fazit

Die Performance der Verknüpfung zwischen SAS® und EXASolution ist maßgeblich von optimaler Programmierung der Abfragen abhängig. Mit optimierten Einstellungen können die Ladezeiten in und aus der Datenbank um ein Vielfaches verkürzt werden. Bei Beachtung einiger grundlegender Regeln können auch die Performance von Table-Joins erheblich verbessert werden. Dies erfordert tiefergehendes Wissen über die Eigenschaften der EXASOL-Datenbank und der Mechanismen, die SAS® für die Bereitstellung von SQL-Abfragen verwendet.

Kontaktieren Sie uns //

mayato GmbH
Am Borsigturm 9
13507 Berlin

info@mayato.com

+49 / 30 4174 4270 10