Effizientes Arbeiten mit relationalen Datenbanken unter SAS - Mit SASTRACE zu mehr Performance -

Philipp Krüger
Versicherungskammer Bayern
Maximilianstraße 53
80530 München
philipp.krueger@vkb.de

Zusammenfassung

Moderne Business Intelligence (BI) Tools zeichnen sich durch eine hohe Integrationsfähigkeit aus, die es dem Nutzer ermöglicht, ohne tiefergehendes Wissen beliebige Datenquellen und Systeme zu integrieren. Zugleich sind moderne BI-Landschaften aber von einer zunehmenden Komplexität gekennzeichnet. Aufgrund dieser lohnt es sich, einen Blick hinter die „Kulissen" moderner BI-Tools zu werfen und sich damit zu beschäftigen, wie diese arbeiten und wie diese die Aufgaben auf die beteiligten Systeme verteilen. Denn dies ist selten optimal, so dass sich hieraus oft erhebliche Effizienzpotenziale ergeben, die es zu nutzen gilt. Ansätze hierzu sollen in diesem Aufsatz aufgezeigt und ein tiefergehendes Verständnis für die Arbeitsweise moderner BI-Tools erreicht werden.

Schlüsselwörter: PROC SQL, SQL Prozedur, DB2, Oracle, SASTRACE, BI, Data Warehouse

1 Die Business Intelligence Landschaft im Wandel

In den vergangenen Jahren ist in vielen Unternehmen ein verstärkter Trend zu erkennen, die dispositive Datenhaltung flexibler und effizienter zu organisieren. Dabei werden oft einfache, flache Datensätze in Form eines Datenfiles, der turnusmäßig aus den operativen Systemen extrahiert wird, durch ein komplexes Data Warehouse auf Basis einer relationalen Datenbank mit normalisiertem Datenmodell und Historisierung ersetzt (Abbildung 1).
Während BI-Tools früher hauptsächlich auf Großrechnersystemen betrieben wurden, wo sie zusammen mit der Datenhaltung eine Einheit bildeten, sind heutige BI-Landschaften meist wesentlich heterogener. Ein eigenständiges Datenbanksystem (DB2, Oracle, etc.) bildet auf der einen Seite oftmals den Kern der Datenhaltung, während auf der anderen Seite ein BI-Tool steht, mittels dessen auf die Daten zugegriffen wird. Dabei wird das BI-Tool oft auf einem eigenständigen System betrieben. Bei kleineren Datenmengen wird heute nur noch ein handelsüblicher PC benötigt, bei größeren Datenvolumen werden hingegen BI-Tools auf UNIX- oder Hostbasis eingesetzt (Abbildung 2).

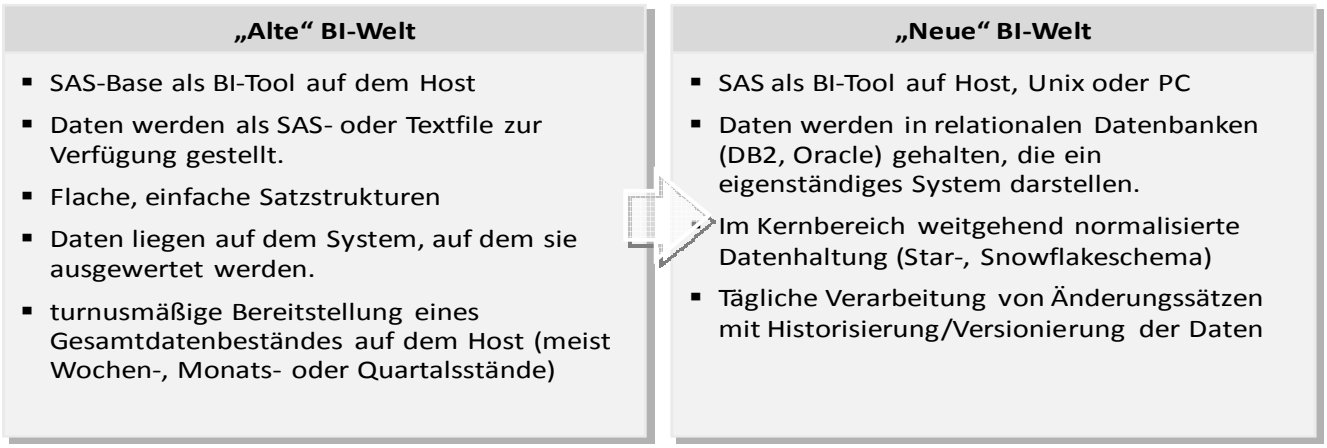

Abbildung 1

Durch die Integrationsfähigkeit der BI-Tools wird dem Anwender ermöglicht, beliebige Datenquellen - wie z.B. Datenbanken - ohne große Aufwände in das BI-Tool einzubinden. Der Nutzer benötigt zunächst kein tiefergehendes Datenbank Know-how um die dort abgelegten Daten für Analysezwecke nutzen zu können. Auch wenn sich die Datenbank scheinbar einfach in das BI-Tool integrieren lässt, ist die Komplexität des neuen Gesamtsystems unweit größer als beim einfachen Zugriff auf einen flachen Datensatz in Form eines Datenfiles.

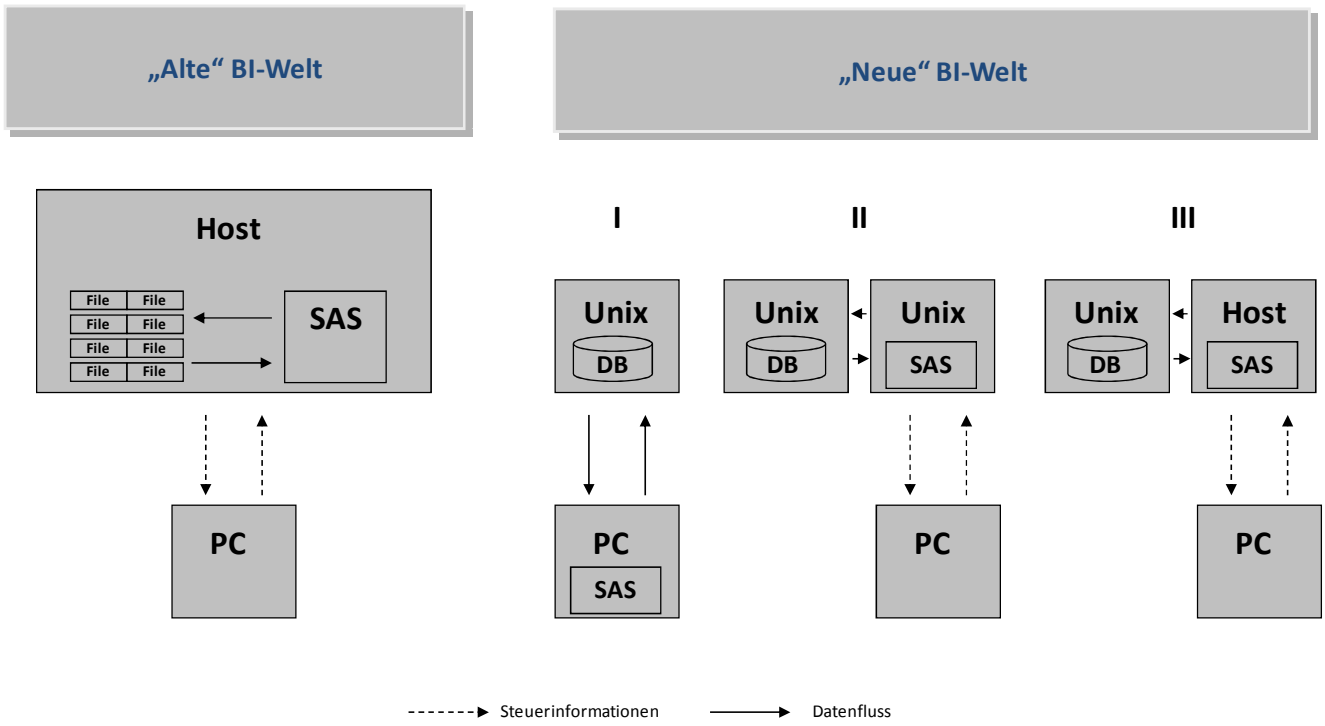

Abbildung 2

Denn moderne Datenbanksysteme sind heute in der Lage, eine Vielzahl von transformatorischen und analytischen Aufgaben selbst zu übernehmen, die früher den BI-Tools vorbehalten waren.

Daher stellt sich die Frage, welche Aufgaben auf welches System in einer BI-Landschaft verteilt werden sollen.
Ohne ein tieferes Verständnis vom Zusammenspiel der Systeme wird der Nutzer schnell zu dem Schluss kommen, dass die Performance des Gesamtsystems zu wünschen übrig lässt. Dies liegt nicht selten an der suboptimalen Verteilung der Aufgaben zwischen den Systemen. Durch die geschickte Verteilung der Aufgaben auf die einzelnen Systeme lassen sich erhebliche Performancesteigerungen erzielen. Dieser integrierte Optimierungsansatz wird oft zu Gunsten partieller Optimierungen an den Einzelsystemen vernachlässigt.

2 Das Zusammenspiel der Systeme

Das folgende Kapitel zeigt an konkreten Beispielen, wie moderne BI-Tools das Zusammenspiel der verschiedenen Systeme koordinieren und welche Optimierungsansätze sich daraus ergeben.

2.1 Konfiguration des Testsystems

Das Testsystem besteht aus einer relationalen DB2-Datenbank und einem Server mit SAS Base. Der SAS-Server ist dabei mit der Datenbank über SAS-Connect (LIBNAME-Statement) verbunden. Gesteuert wird der SAS-Server remote über einen PC mit PC-SAS. Die Konstellation entspricht der in Abbildung 2 Typ II dargestellten Form.

Die Datenbank:
- AIX
- DB2 9.5.4
- Historisierte Vertragstabelle dw_lv_ver_vvs mit 2.4 Mio. Zeilen und 41 Spalten.
- Testdatensatz mit 111 Zeilen und 41 Spalten

Der BI-Server:
- AIX 5.3
- SAS Base 9.1.3 Service Pack 4
- Testdatensatz mit 111 Zeilen und 41 Spalten

2.2 SASTRACE als Ansatz für mehr Performance

Mit der System Option SASTRACE lässt sich die Kommunikation zwischen SAS Base auf dem BI-Server und der Datenbank offenlegen (Abbildung 3). Dies ist Voraussetzung um zu verstehen, wie einzelne Arbeitsschritte auf die verschiedenen Systeme verteilt werden und um daraus Optimierungsansätze abzuleiten.

SASTRACE = System Option

Syntax
SASTRACE= ',,,d' | ',,d,' | 'd,' | ',,,s' | ',,,sa' | ',,t,'

- **´,,,d´**
  gibt an, dass alle SQL-Statements, die zur Datenbank gesendet werden, im Log angezeigt werden.
- **´,,d,´**
  gibt an, dass alle Routinenaufrufe im Log angezeigt werden.
- **´d,´**
  gibt an, dass alle Datenbankaufrufe, wie z.B. API-Aufrufe im Log angezeigt werden.
- **´,,,s´**
  generiert eine Zusammenfassung von Laufzeitinformationen, die durch Aufrufe verursacht wurden.
- **´,,,sa´**
  generiert Laufzeitinformationen von Aufrufen und eine Zusammenfassung.
- **´,,t,´**
  gibt an, dass alle Threads im Log angezeigt werden.

Abbildung 3

2.3 Das Zusammenspiel der Systeme anhand konkreter Beispiele

Anhand von 2 Beispielen soll zum einen demonstriert werden, wie sich die Systeme in einem konkreten Fall verhalten. Zum anderen soll damit aufgezeigt werden, wie man die System Option SASTRACE als generisches Instrument einsetzen kann, um die Performance des Gesamtsystems zu verbessern.

2.3.1 Kopieren einer Tabelle

Als erstes Beispiel dient das Kopieren einer Tabelle. Dabei liegt die Tabelle auf der Datenbank. Das Kopieren wird über SAS-Base angestoßen. Die Zieltabelle soll wiederum auf der Datenbank liegen. Will man das Beispiel komplexer und praxisnäher gestalten, könnte man beispielsweise noch auf den aktuell gültigen Satz einer Vertragsversion einschränken. Hierauf wird aber bewusst verzichtet.
Das Kopieren der Tabelle lässt sich - wie in Abbildung 4 dargestellt - in SAS-BASE auf zweierlei Arten erreichen: Durch einen DATA-Step oder mittels PROC SQL. Beide Vorgehensarten führen zu einem identischen Ergebnis, nämlich zu einem exakten Abbild der Originaltabelle auf der Datenbank. In beiden Fällen dauert das Kopieren der 2.4 Mio. Datensätze ca. 6 Minuten. Am SQL-Code des PROC SQL erkennt man, dass es sich hierbei nicht um proprietäres DB2-SQL handelt, sondern um ein generisches SAS-SQL, das von SAS-BASE zunächst in ein datenbankspezifisches SQL übersetzt werden muss, denn DB2-SQL kennt kein CREATE TABLE as SELECT.
Wird das SQL-Statement zum Kopieren der Tabelle nicht in SAS-Base sondern direkt auf der Datenbank abgesetzt, verringert sich die Verarbeitungszeit auf 30 Sekunden, was einer Zeitersparnis von ca. 90 % entspricht.

DATA-Step

```
DATA dwhv99.test2;
SET dwhv01.dw_lv_ver_vvs;
RUN;
```

PROC SQL

```
PROC SQL;
CREATE TABLE dwhv99.test1 AS
SELECT * FROM
  dwhv01.dw_lv_ver_vvs;
```

→ Laufzeit der Programme in beiden Fällen ca. 6 Minuten

SQL direkt auf der DB2

```
CREATE TABLE dwhv99.test1 AS (SELECT * FROM dwhv01.dw_lv_ver_vvs)
  WITH NO DATA;
INSERT INTO dwhv99.test1 (SELECT * FROM dwhv01.dw_lv_ver_vvs);
```

→ Laufzeit des Programms ca. 30 Sekunden

Abbildung 4

Um klären zu können, wie es zu diesen Laufzeitabweichungen kommt, muss man verstehen, was das Ausführen des DATA-Steps bzw. des PROC SQLs bewirkt.
Hierzu wird vor der erneuten Ausführung des Programmcodes die System Option SASTRACE mit dem Parameter `„„d` gesetzt. Dies bewirkt, dass die gesamte Kommunikation mit der Datenbank im Logfile ausgegeben wird.

Wie der erste Teil des Logfiles in Abbildung 5 zeigt, wird zunächst die gesamte Tabelle mittels SELECT-Statements auf den BI-Server übertragen.
Daraufhin wird, wie in Abbildung 6 ersichtlich, auf der Datenbank die Zieltabelle angelegt.

```
48   proc sql;
49   create table dwhv99.test1 as
50   select * from dwhv99.testdatensatz; 102 1260809433 du_prep 0 SQL
DB2_22: Prepared: 103 1260809433 du_prep 0 SQL
SELECT * FROM DWHV99.TESTDATENSATZ FOR READ ONLY  104 1260809433 du_prep 0 SQL
  105 1260809433 du_prep 0 SQL
DB2: COMMIT performed on connection 1. 106 1260809433 du_comm 0 SQL
  107 1260809433 du_prep 0 SQL
DB2_23: Prepared: 108 1260809433 du_prep 0 SQL
SELECT * FROM DWHV99.TEST1 WHERE 0=1  FOR READ ONLY  109 1260809433 du_prep 0
SQL
  110 1260809433 du_prep 0 SQL
DB2: COMMIT performed on connection 2. 111 1260809433 du_comm 0 SQL
DB2: AUTOCOMMIT is NO for connection 3 112 1260809433 xoopen 0 SQL
NOTE: SAS variable labels, formats, and lengths are not written to DBMS tables.
  113 1260809433 du_exec 0 SQL
DB2_24: Executed: 114 1260809433 du_exec 0 SQL
Prepared statement DB2_22 115 1260809433 du_exec 0 SQL
  116 1260809433 du_exec 0 SQL
  117 1260809433 du_execi 0 SQL
```

Abbildung 5

```
DB2_25: Executed: 118 1260809433 du_execi 0 SQL
CREATE TABLE DWHV99.TEST1 (DW_LV_VER_VVS_SID NUMERIC(21,0),DW_MANDANT_ID
VARCHAR(5),DW_PRD_GESELL_ID VARCHAR(10),DW_VERTRAG_SID NUMERIC(21,0),DW_LVID
VARCHAR(40),DW_LVID_EXT VARCHAR(40),DW_LV_HVGRUPPE_SID NUMERIC(21,0),DW_BEARBID
NUMERIC(12,0),DW_SCHRITT_ID NUMERIC(12,0),DW_BEARBGRDID
NUMERIC(12,0),DW_LV_PRODINFO_SID NUMERIC(21,0),DW_PDID VARCHAR(40),DW_GENERATION
VARCHAR(20),DW_ZINS_BDEP_ID NUMERIC(18,8),DW_BEARBID_ABG NUMERIC(12,0),
DW_BETRAG_BDEP NUMERIC(18,2),DW_WAEHRUNG_ID NUMERIC(12,0),DW_LVBEGINN_DAT DATE,
DW_LVABLAUF_DAT DATE,DW_BTR_DIFF_MIG NUMERIC(18,2), DW_LV_JAHRTAG DATE,
DW_LV_STATUS_ID NUMERIC(12,0),DW_LET_SWITCH DATE,DW_DYNERH_ABS
NUMERIC(18,2),DW_ABL_BDEP DATE,DW_VEREINB_BDEP_ID NUMERIC(12,0),DW_LETZT_NOV_DAT
DATE,DW_ABWFAELL NUMERIC(12,0),DW_VN_VP_KZ NUMERIC(11,0),DW_GE_GS_CD VARCHAR(4),
DW_BAT DATE,DW_BAT_ENDE DATE,DW_WIT DATE,DW_WIT_ENDE DATE,DW_STOBEARBID
NUMERIC(12,0),DW_STOBAT DATE,DW_STOWIT DATE,DW_ADMERSTELLZEIT TIMESTAMP,
DW_ADMLADELAUF_ID NUMERIC(12,0),DW_ADMJOB_ID VARCHAR(128),DW_ADMSRC_CD
VARCHAR(3)) 119 1260809433 du_execi 0 SQL
 120 1260809433 du_execi 0 SQL
DB2: COMMIT performed on connection 3. 121 1260809433 du_comm 0 SQL
  122 1260809433 du_prep 0 SQL
```

Abbildung 6

```
DB2_26: Prepared: 123 1260809433 du_prep 0 SQL
INSERT INTO DWHV99.TEST1
(DW_LV_VER_VVS_SID,DW_MANDANT_ID,DW_PRD_GESELL_ID,DW_VERTRAG_SID,DW_LVID,DW_LVID
_EXT,DW_LV_HVGRUPPE_SID, DW_BEARBID, DW_SCHRITT_ID,DW_BEARBGRDID,
DW_LV_PRODINFO_SID
,DW_PDID,DW_GENERATION,DW_ZINS_BDEP_ID,DW_BEARBID_ABG,DW_BETRAG_BDEP,DW_WAEHRUNG
_ID,DW_LVBEGINN_DAT,DW_LVABLAUF_DAT,DW_BTR_DIFF_MIG,DW_LV_JAHRTAG,DW_LV_STATUS_I
D,
DW_LET_SWITCH,DW_DYNERH_ABS,DW_ABL_BDEP,DW_VEREINB_BDEP_ID,DW_LETZT_NOV_DAT,
DW_ABWFAELL,DW_VN_VP_KZ,DW_GE_GS_CD,DW_BAT,DW_BAT_ENDE,DW_WIT,DW_WIT_ENDE,
DW_STOBEARBID,DW_STOBAT,DW_STOWIT,DW_ADMERSTELLZEIT,DW_ADMLADELAUF_ID,DW_ADMJOB_
ID,DW_ADMSRC_CD)  VALUES ( ? , ? ,? , ? , ? , ? , ? , ? , ? , ? , ? , ? , ? , ?
, ? , ? , ? , ? , ? , ? , ? , ? , ? , ? , ? , ? , ? , ?, ? , ? , ? , ? , ? , ? ,
? , ? , ? , ? , ? , ? , ? ) 124 1260809433 du_prep 0 SQL
  125 1260809433 du_prep 0 SQL
  126 1260809433 du_exec 0 SQL
DB2_27: Executed: 127 1260809433 du_exec 0 SQL
Prepared statement DB2_26 128 1260809433 du_exec 0 SQL
  129 1260809433 du_exec 0 SQL
  130 1260809433 du_exec 0 SQL
DB2_28: Executed: 131 1260809433 du_exec 0 SQL
Prepared statement DB2_26 132 1260809433 du_exec 0 SQL
  133 1260809433 du_exec 0 SQL
  134 1260809433 du_exec 0 SQL
DB2_29: Executed: 135 1260809433 du_exec 0 SQL
Prepared statement DB2_26 136 1260809433 du_exec 0 SQL
  137 1260809433 du_exec 0 SQL
  138 1260809433 du_exec 0 SQL
DB2_30: Executed: 139 1260809433 du_exec 0 SQL
Prepared statement DB2_26 140 1260809433 du_exec 0 SQL
  141 1260809433 du_exec 0 SQL
DB2: COMMIT performed on connection 3. 142 1260809433 du_comm 0 SQL
DB2: COMMIT performed on connection 3. 143 1260809433 du_comm 0 SQL
NOTE: Table DWHV99.TEST1 created, with 111 rows and 41 columns.
```

Abbildung 7

In den letzten beiden Schritten (Abbildung 7) wird dann die Zieltabelle vom BI-Server mittels INSERT-Statement befüllt.

Dass dieses Vorgehen suboptimal ist, ist offensichtlich. Beherrschen doch moderne Datenbanksysteme diese Arten von Operationen selbst. Ein Umweg der Daten über den BI-Server ist daher unnötig. Hier wäre SAS-Base aufgefordert, nur eine steuernde Funktion zu übernehmen und nicht die Daten selbst physisch zwischen den Systemen auszutauschen.

Mit Hilfe der Option DBIDIRECTEXEC [1] kann SAS angewiesen werden, bei CREATE TABLE und DELETE-Anweisungen möglichst die gesamte Aktivität auf die Datenbank zu verlagern.

DATA-Step

```
OPTIONS DBIDIRECTEXEC;
DATA dwhv99.test2;
SET dwhv01.dw_lv_ver_vvs;
RUN;
```

→ Laufzeit ca. 6 Minuten

PROC SQL

```
OPTIONS DBIDIRECTEXEC;
PROC SQL;
CREATE TABLE dwhv99.test1 AS
SELECT * FROM
  dwhv01.dw_lv_ver_vvs;
```

→ Laufzeit ca. 30 Sekunden

Abbildung 8

Wie Abbildung 8 zeigt, macht es nun einen Unterschied, ob ein DATA-Step oder eine PROC SQL-Anweisung verwendet wird. Die Option DBIDIRECTEXEC wirkt offensichtlich nur bei PROC SQL-Anweisungen. Bei DATA-Steps bleibt die Herangehensweise identisch, so dass hier auf das erneute Ausgeben des Logfiles verzichtet werden soll. Weit interessanter ist die Vorgehensweise, die sich aus dem PROC SQL ergibt.

```
15   options dbidirectexec;
16   options sastrace =',,,d' sastraceloc=saslog;
17   proc sql;
18   create table dwhv99.test1 as
19   select * from dwhv99.testdatensatz;
DB2: AUTOCOMMIT turned ON for connection id 1 0 1260864131 xoopen 0 SQL
  1 1260864131 du_prep 0 SQL
DB2_1: Prepared: 2 1260864131 du_prep 0 SQL
SELECT * FROM DWHV99.TEST1 FOR READ ONLY  3 1260864131 du_prep 0 SQL
  4 1260864131 du_prep 0 SQL
DB2: ROLLBACK performed on connection 1. 5 1260864131 du_comm 0 SQL
  6 1260864131 du_prep 0 SQL
DB2_2: Prepared: 7 1260864131 du_prep 0 SQL
SELECT * FROM DWHV99.TESTDATENSATZ FOR READ ONLY  8 1260864131 du_prep 0 SQL
  9 1260864131 du_prep 0 SQL
DB2: COMMIT performed on connection 1. 10 1260864131 du_comm 0 SQL
DB2: AUTOCOMMIT is NO for connection 2 11 1260864131 sqlpnas 0 SQL
DB2: AUTOCOMMIT turned ON for connection id 2 12 1260864131 sqlpnas 0 SQL
  13 1260864131 du_execi 0 SQL
```

Abbildung 9

Wie dem ersten Teil des Logfiles (Abbildung 9) entnommen werden kann, wird im Gegensatz zur ursprünglichen Herangehensweise die Tabelle nicht mehr auf den BI-Server übertragen. Vielmehr wird nur die Struktur der Tabelle mittels Prepare ermittelt.
Da das DB2-SQL keine „CREATE TABLE as SELECT"-Anweisung kennt, wird zunächst in einem eigenen Schritt die Zieltabelle auf der Datenbank angelegt (Abbildung 10).

```
DB2_3: Executed: 14 1260864131 du_execi 0 SQL
CREATE TABLE DWHV99.test1 as ( select DWHV99.testdatensatz."DW_LV_VER_VVS_SID",
DWHV99.testdatensatz."DW_MANDANT_ID", DWHV99.testdatensatz."DW_PRD_GESELL_ID",
DWHV99.testdatensatz."DW_VERTRAG_SID", DWHV99.testdatensatz."DW_LVID",
DWHV99.testdatensatz."DW_LVID_EXT", DWHV99.testdatensatz."DW_LV_HVGRUPPE_SID",
DWHV99.testdatensatz."DW_BEARBID", DWHV99.testdatensatz."DW_SCHRITT_ID",
DWHV99.testdatensatz."DW_BEARBGRDID", DWHV99.testdatensatz."DW_LV_PRODINFO_SID",
DWHV99.testdatensatz."DW_PDID", DWHV99.testdatensatz."DW_GENERATION",
DWHV99.testdatensatz."DW_ZINS_BDEP_ID", DWHV99.testdatensatz."DW_BEARBID_ABG",
DWHV99.testdatensatz."DW_BETRAG_BDEP", DWHV99.testdatensatz."DW_WAEHRUNG_ID",
DWHV99.testdatensatz."DW_LVBEGINN_DAT", DWHV99.testdatensatz."DW_LVABLAUF_DAT",
DWHV99.testdatensatz."DW_BTR_DIFF_MIG", DWHV99.testdatensatz."DW_LV_JAHRTAG",
DWHV99.testdatensatz."DW_LV_STATUS_ID", DWHV99.testdatensatz."DW_LET_SWITCH",
DWHV99.testdatensatz."DW_DYNERH_ABS", DWHV99.testdatensatz."DW_ABL_BDEP",
DWHV99.testdatensatz."DW_VEREINB_BDEP_ID",
DWHV99.testdatensatz."DW_LETZT_NOV_DAT",
DWHV99.testdatensatz."DW_ABWFAELL", DWHV99.testdatensatz."DW_VN_VP_KZ",
DWHV99.testdatensatz."DW_GE_GS_CD", DWHV99.testdatensatz."DW_BAT",
DWHV99.testdatensatz."DW_BAT_ENDE", DWHV99.testdatensatz."DW_WIT",
DWHV99.testdatensatz."DW_WIT_ENDE", DWHV99.testdatensatz."DW_STOBEARBID",
DWHV99.testdatensatz."DW_STOBAT", DWHV99.testdatensatz."DW_STOWIT",
DWHV99.testdatensatz."DW_ADMERSTELLZEIT",
DWHV99.testdatensatz."DW_ADMLADELAUF_ID",
DWHV99.testdatensatz."DW_ADMJOB_ID", DWHV99.testdatensatz."DW_ADMSRC_CD" from
DWHV99.TESTDATENSATZ ) DEFINITION ONLY  15 1260864131 du_execi 0 SQL
  16 1260864131 du_execi 0 SQL
DB2: COMMIT performed on connection 2. 17 1260864131 du_comm 0 SQL
  18 1260864131 du_execi 0 SQL
```

Abbildung 10

Daraufhin wird die Zieltabelle direkt aus der Quelltabelle der Datenbank befüllt (Abbildung 11). Somit ergibt sich ein analoges Vorgehen zum direkten Abschicken des Statements auf der Datenbank (Abbildung 4) mit identischer Ausführungsdauer.

```
DB2_4: Executed: 19 1260864131 du_execi 0 SQL
INSERT INTO DWHV99.test1 select DWHV99.testdatensatz."DW_LV_VER_VVS_SID",
DWHV99.testdatensatz."DW_MANDANT_ID", DWHV99.testdatensatz."DW_PRD_GESELL_ID",
DWHV99.testdatensatz."DW_VERTRAG_SID", DWHV99.testdatensatz."DW_LVID",
DWHV99.testdatensatz."DW_LVID_EXT", DWHV99.testdatensatz."DW_LV_HVGRUPPE_SID",
DWHV99.testdatensatz."DW_BEARBID", DWHV99.testdatensatz."DW_SCHRITT_ID",
DWHV99.testdatensatz."DW_BEARBGRDID", DWHV99.testdatensatz."DW_LV_PRODINFO_SID",
DWHV99.testdatensatz."DW_PDID", DWHV99.testdatensatz."DW_GENERATION",
DWHV99.testdatensatz."DW_ZINS_BDEP_ID", DWHV99.testdatensatz."DW_BEARBID_ABG",
DWHV99.testdatensatz."DW_BETRAG_BDEP", DWHV99.testdatensatz."DW_WAEHRUNG_ID",
DWHV99.testdatensatz."DW_LVBEGINN_DAT", DWHV99.testdatensatz."DW_LVABLAUF_DAT",
DWHV99.testdatensatz."DW_BTR_DIFF_MIG", DWHV99.testdatensatz."DW_LV_JAHRTAG",
DWHV99.testdatensatz."DW_LV_STATUS_ID", DWHV99.testdatensatz."DW_LET_SWITCH",
DWHV99.testdatensatz."DW_DYNERH_ABS", DWHV99.testdatensatz."DW_ABL_BDEP",
DWHV99.testdatensatz."DW_VEREINB_BDEP_ID",
DWHV99.testdatensatz."DW_LETZT_NOV_DAT",
DWHV99.testdatensatz."DW_ABWFAELL", DWHV99.testdatensatz."DW_VN_VP_KZ",
DWHV99.testdatensatz."DW_GE_GS_CD", DWHV99.testdatensatz."DW_BAT",
DWHV99.testdatensatz."DW_BAT_ENDE", DWHV99.testdatensatz."DW_WIT",
DWHV99.testdatensatz."DW_WIT_ENDE", DWHV99.testdatensatz."DW_STOBEARBID",
DWHV99.testdatensatz."DW_STOBAT", DWHV99.testdatensatz."DW_STOWIT",
DWHV99.testdatensatz."DW_ADMERSTELLZEIT",
DWHV99.testdatensatz."DW_ADMLADELAUF_ID",
DWHV99.testdatensatz."DW_ADMJOB_ID", DWHV99.testdatensatz."DW_ADMSRC_CD" from
DWHV99.TESTDATENSATZ 20 1260864131 du_execi 0 SQL
  21 1260864131 du execi 0 SQL
```

Abbildung 11

2.3.2 Join mit 2 Tabellen

Als zweites Beispiel dient der (Inner-) Join zweier Tabellen [2]. Dabei liegt eine große Tabelle mit 2.4 Mio. Datensätzen auf der Datenbank und eine kleinere mit 111 Datensätzen auf dem BI-Server. Als Ergebnis soll eine angereicherte Datei mit 111 Datensätzen auf dem BI-Server entstehen (Abbildung 12).

PROC SQL

```
proc sql;
create table test as
select
 dw_lv_ver_vvs.*
from
 dwhv01.dw_lv_ver_vvs inner join testdatensatz  on
  dw_lv_ver_vvs.dw_lv_ver_vvs_sid=testdatensatz.dw_lv_ver_vvs_sid;
```

→ Laufzeit des Programms ca. 2 ½ Minuten

Abbildung 12

Auch hier sollen durch die Logfileanalyse mögliche Effizienzpotenziale gehoben werden. Obwohl von der Tabelle, die sich auf der Datenbank befindet, nur die 111 Datensätzen der anderen Tabelle, die sich auf dem BI-Server befindet, benötigt werden, wird die gesamte Tabelle auf den BI-Server übertragen (Abbildung 13).

```
188  proc sql;
189  create table test1 as
190  select
191   dw_lv_ver_vvs.*
192  from
193              dwhv01.dw_lv_ver_vvs
194   inner join testdatensatz  on
dw_lv_ver_vvs.dw_lv_ver_vvs_sid=testdatensatz.dw_lv_ver_vvs_sid
194! ;
  558 1262706844 du_prep 0 SQL
DB2_136: Prepared: 559 1262706844 du_prep 0 SQL
SELECT * FROM DWHV01.DW_LV_VER_VVS FOR READ ONLY  560 1262706844 du_prep 0 SQL
  561 1262706844 du_prep 0 SQL
DB2: COMMIT performed on connection 0. 562 1262706844 du_comm 0 SQL
  563 1262706844 du_exec 0 SQL
DB2_137: Executed: 564 1262706844 du_exec 0 SQL
Prepared statement DB2_136 565 1262706844 du_exec 0 SQL
  566 1262706844 du_exec 0 SQL
NOTE: Table WORK.TEST1 created, with 111 rows and 41 columns.
```

Abbildung 13

Dass auch diese Herangehensweise suboptimal ist, ist offensichtlich. Mit Hilfe der Data Set Option DBKEY wird versucht, die Abfrage mittels eines Schlüssels zu optimieren (Abbildung 14).

PROC SQL

```
proc sql;
create table test1 as
select
 dw_lv_ver_vvs.*
from
            dwhv01.dw_lv_ver_vvs  (dbkey=(dw_lv_ver_vvs_sid))
 inner join testdatensatz  on
  dw_lv_ver_vvs.dw_lv_ver_vvs_sid=testdatensatz.dw_lv_ver_vvs_sid;
```

→ Laufzeit ca. 1 Sekunde

Abbildung 14

Wie der erste Teil der Ausgabe des Logfiles (Abbildung 15) zeigt, wird das SELECT-Statement jetzt nur noch vorbereitet, aber nicht mehr ausgeführt. Somit wird nicht mehr die gesamte Tabelle übertragen, sondern nur die Tabellenstruktur ermittelt.

```
199  proc sql;
200  create table test2 as
201  select
202   dw_lv_ver_vvs.*
203  from
204              dwhv01.dw_lv_ver_vvs  (dbkey=(dw_lv_ver_vvs_sid))
205   inner join testdatensatz  on
dw_lv_ver_vvs.dw_lv_ver_vvs_sid=testdatensatz.dw_lv_ver_vvs_sid
205! ;
  567 1262709949 du_prep 0 SQL
DB2_138: Prepared: 568 1262709949 du_prep 0 SQL
SELECT * FROM DWHV01.DW_LV_VER_VVS FOR READ ONLY  569 1262709949 du_prep 0 SQL
  570 1262709949 du_prep 0 SQL
DB2: COMMIT performed on connection 0. 571 1262709949 du_comm 0 SQL
  572 1262709949 du_prep 0 SQL
```

Abbildung 15

Im zweiten Schritt (Abbildung 16) wird zunächst das Select-Statement für die Abfrage der benötigten Datensätze vorbereitet, bevor im dritten Schritt dann die benötigten Datensätze einzeln von der Datenbank abgefragt werden.

```
DB2_139: Prepared: 573 1262709949 du_prep 0 SQL
SELECT  "DW_LV_VER_VVS_SID", "DW_MANDANT_ID", "DW_PRD_GESELL_ID",
"DW_VERTRAG_SID",
"DW_LVID","DW_LVID_EXT","DW_LV_HVGRUPPE_SID","DW_BEARBID","DW_SCHRITT_ID",
"DW_BEARBGRDID","DW_LV_PRODINFO_SID","DW_PDID","DW_GENERATION","DW_ZINS_BDEP_ID"
, "DW_BEARBID_ABG","DW_BETRAG_BDEP","DW_WAEHRUNG_ID","DW_LVBEGINN_DAT",
"DW_LVABLAUF_DAT","DW_BTR_DIFF_MIG","DW_LV_JAHRTAG","DW_LV_STATUS_ID",
"DW_LET_SWITCH", "DW_DYNERH_ABS", "DW_ABL_BDEP","DW_VEREINB_BDEP_ID",
"DW_LETZT_NOV_DAT", "DW_ABWFAELL", "DW_VN_VP_KZ", "DW_GE_GS_CD", "DW_BAT",
"DW_BAT_ENDE", "DW_WIT", "DW_WIT_ENDE", "DW_STOBEARBID", "DW_STOBAT",
"DW_STOWIT",
"DW_ADMERSTELLZEIT", "DW_ADMLADELAUF_ID", "DW_ADMJOB_ID", "DW_ADMSRC_CD"  FROM
DWHV01.DW_LV_VER_VVS  WHERE ((((("DW_LV_VER_VVS_SID"= ? )))) FOR READ ONLY  574
1262709949 du_prep
0 SQL
  575 1262709949 du_prep 0 SQL
  576 1262709949 du_exec 0 SQL
DB2_140: Executed: 577 1262709949 du_exec 0 SQL
Prepared statement DB2_139 578 1262709949 du_exec 0 SQL
  579 1262709949 du_exec 0 SQL
  580 1262709949 du_exec 0 SQL
DB2_141: Executed: 581 1262709949 du_exec 0 SQL
Prepared statement DB2_139 582 1262709949 du_exec 0 SQL
  583 1262709949 du_exec 0 SQL
  584 1262709949 du_exec 0 SQL
DB2_142: Executed: 585 1262709949 du_exec 0 SQL
Prepared statement DB2_139 586 1262709949 du_exec 0 SQL
  587 1262709949 du_exec 0 SQL
  588 1262709949 du_exec 0 SQL
```

Abbildung 16

Wie schon im ersten Beispiel bewirkt auch diese Maßnahme einen enormen Performancegewinn.

3 Fazit

Wie die Beispiele gezeigt haben, liegt in der geschickten Steuerung der Systeme ein enormes Potential zur Performancesteigerung. Dieses wird allzu oft vernachlässigt. Optimiert die IT primär die Einzelsysteme (z.B. durch Anlegen von Indizies auf der Datenbank), die oft auch noch von verschiedenen Abteilungen betreut werden, hat der Anwender die Analytik im Blick. Durch diese Aufgabenteilung geht oft der Blick für das Gesamtsystem verloren. Die optimale Verteilung der Aufgaben auf die Systeme muss letztlich vom Anwender kommen, der das System für analytische Zwecke nutzt.

Denn für jede Analyse ergibt sich eine neue optimale Verteilung der Aufgaben auf die Systeme. SASTRACE hilft dabei, die geeigneten Maßnahmen zu ergreifen.
Als Faustregel lässt sich festhalten, dass Daten möglichst dort verarbeitet werden sollten, wo sie auch physisch gehalten werden.
Da dies heutzutage meist eine Datenbank ist, kann es sinnvoll sein, einen großen Teil der Arbeit direkt auf der Datenbank auszuführen, indem man mittels Datenbanktools direkt auf die Datenbank zugreift. Somit vermeidet man ständige Verteilungsprobleme und die Notwendigkeit, sich damit beschäftigen zu müssen. Dies setzt auf der anderen Seite Datenbank Know-how voraus, was dem Ansatz moderner BI-Tools entgegensteht.

Literatur

[1] Macro Variables and System Options for Relational Databases: DBIDIRECTEXEC
http://support.sas.com/documentation/cdl/en/acreldb/61890/HTML/default/a003228411.htm#

[2] Garth, W. Helf: Joining SAS and DBMS Tables Efficiently
http://www2.sas.com/proceedings/sugi26/p127-26.pdf