
mitp
Cornel Brücher
Thomas Glörfeld
Microsoft SQL Thinking
Vom Problem zum SQL-Statement
Für SQL Server 2012

Kapitel 1

# Das Problem mit dem Problem

Bevor man sich auf den Weg vom Problem zum SQL-Statement machen kann, muss man zuerst das Problem genau kennen. Da Zitate sich in einer Einleitung immer gut machen und wir zufällig ein passendes gefunden haben, lassen wir an dieser Stelle den berühmten Strategen Sun Tsu zu Wort kommen: »Wenn du den Feind und dich selbst kennst, brauchst du den Ausgang von hundert Schlachten nicht zu fürchten.« Etwas zeitgemäßer und in die IT übertragen: »Wenn du die Fragestellung und deine Daten kennst, brauchst du das Ergebnis von hundert Abfragen nicht zu fürchten.«

Da fängt es an, das Problem mit dem Problem. Es ist schon Tradition, Anfragen an die IT unpräzise, wolkig oder global-galaktisch zu stellen. Die IT kann Gedanken lesen und liefert automatisch das richtige Ergebnis. Wie das in der Praxis tatsächlich aussieht, hat Douglas Adams, der wohl bekannteste Experte für unpräzise Fragestellungen und deren Auswirkungen, in seinen Werken ausführlich dargestellt. Auf die Frage nach dem Projekt, dem Budget und dem ganzen Rest können Sie einfach mit »42%« antworten. Das »%« ist bei Betriebswirten sehr beliebt und unterstreicht die Glaubwürdigkeit Ihrer Antwort.

Das Problem mit dem Problem ist also die präzise Formulierung desselben. Allzu oft schaffen es unpräzise Fragestellungen durch alle Konzeptstufen hindurch bis zum Entwickler, der sich dann die fehlenden Anforderungsdetails nach bestem Wissen und Gewissen zusammenreimen muss. Das Ergebnis wird dann klassifiziert als »works as designed«, eine vornehme Umschreibung für »Das Ergebnis kann zwar keiner gebrauchen, aber wir haben geliefert, was bestellt wurde«. An der präzisen Problembeschreibung bzw. Fragestellung führt kein Weg vorbei.

Die zentrale Frage vor jeder Datenbankabfrage lautet:

»Welche Frage sollen die Daten beantworten?«

Stellen Sie diese Frage demjenigen, der Ihre Daten haben will. Wenn niemand sonst verfügbar ist, stellen Sie sich selbst diese Frage, bevor Sie beginnen.

Damit wir aber jetzt zur Sache kommen können, setzen wir präzise Fragestellungen einfach voraus. Wir haben uns das bei den Mathematikern abgeschaut und sagen einfach:

Gegeben sei eine präzise Fragestellung.

## 1.1 Sssickwl?

SQL wird auf Englisch eigentlich als [ɛskju:'ɛl] ausgesprochen, woran sich unser Lektor eisern hält. Unter den englischsprachigen Datenbankexperten ist aber die Aussprache ['si:kwəl] weit verbreitet, die noch auf den Vorgänger SEQUEL zurückgeht. Mit der lässigen Art, den Titel des Buches als »Sssickwl Sssinking« auszusprechen, können Sie sich schon mal mit der Aura des Experten umgeben.

SQL (Structured Query Language) ist die standardisierte Datenzugriffssprache für relationale Datenbanken. SQL unterteilt sich in DDL (Data Definition Language) und DML (Data Manipulation Language). In DDL werden die Datenbankobjekte und -strukturen definiert (zum Beispiel Tabellen angelegt, der lästige Verwaltungskram halt – siehe weiter hinten im Buch). Mit DML werden die Daten eingefügt (INSERT), geändert (UPDATE), abgefragt (SELECT) und gelöscht (DELETE).

In diesem Buch liegt der Schwerpunkt auf DML, also dem Arbeiten mit den Daten selbst, und insbesondere auf der Abfrage der Daten mit SELECT. Sie sollen ein Gefühl für diese Abfragen bekommen, und Sie werden sehen, dass man mit SQL-Abfragen auf einer Datenbank Auswertungen erzeugen kann, die normalerweise mit viel Nach(t)arbeit und einem schwarzen Gürtel in Excel erledigt werden. Lediglich Grafiken erzeugen wir mit SQL nicht, aber für die Lieferung von Grafiken an Betriebswirte gibt es eine weitere einfache Regel: Die Kurve muss immer von links unten nach rechts oben verlaufen.

Das Schlüsselwort für jede Datenbankabfrage ist **SELECT**. Die mit SELECT eingeleitete Datenbankabfrage wird **SELECT-Statement** genannt. Die Grundlage aller SELECT-Statements lässt sich so zusammenfassen:

```
SELECT irgendwas FROM irgendwoher
```

Mit einem SELECT-Statement formulieren Sie nicht, was ein Programm tun soll. SQL ist keine imperative, sondern eine deklarative Programmiersprache. Es lassen sich keine Programmabläufe, Verzweigungen oder Schleifen formulieren. Mit Befehl und Gehorsam funktioniert hier also nichts. Sie beschreiben, was Sie haben wollen, und wo es zu finden ist. Man könnte auch sagen, Sie spielen eine Art »Hol das Stöckchen!« mit der Datenbank. Das erfordert schon die gleiche Präzision wie beim Programmieren. Wenn Sie Ihre Abfrage nicht genau genug formulieren, kann es passieren, dass Sie statt des erwarteten Stöckchens den ganzen Wald bekommen, nur Holzspäne oder gar nichts.

Ein nicht unbedeutender Nebeneffekt unpräziser Abfragen auf einer Produktionsdatenbank ist übrigens eine hohe Systemlast, mit der man die Kollegen und insbesondere den Administrator gegen sich aufbringt.

Sprachlich gesehen ist eine Datenbankabfrage eher ein Befehl als eine Frage. Am Anfang steht aber tatsächlich eine Frage. Irgendjemand in der (betriebswirtschaft-

lichen) Welt da draußen will irgendetwas wissen, das nur Sie herausfinden können, weil die Antwort irgendwo da drin (in der Datenbank) ist. Wir beginnen mal mit einer alltäglichen Frage.

## 1.2 Die ersten Fragen

### 1.2.1 Voraussetzungen

Falls Sie noch über keine Erfahrung mit dem SQL-Server verfügen, empfehlen wir Ihnen an dieser Stelle Anhang A. Dort werden die Installation des SQL-Servers und die ersten Schritte in der Bedienung des SQL Server Management Studios beschrieben sowie das Importieren der Beispieldaten.

Sind die Beispieldaten bereits vorhanden, öffnen Sie im Objekt-Explorer den Ordner *Datenbanken*, danach mit Rechtsklick auf die Datenbank *SQL-Thinking* das Kontextmenü und wählen dort den Eintrag *Neue Abfrage* aus. In dem so geöffneten Fenster führen wir unsere SQL-Abfragen aus.

### 1.2.2 Welches Datum haben wir heute?

Nicht auf den Kalender schauen, die Datenbank weiß alles. Wir benötigen lediglich einen Funktionsaufruf. Das Datum erfährt man über die SQL-Funktion GETDATE(). SQL-Funktionen verhalten sich wie Funktionen in Programmiersprachen. Sie rufen die Funktion im SQL-Statement in einer Spalte auf, und bei der Ausgabe des SELECT-Statements steht an der entsprechenden Stelle das Ergebnis der Funktion, der sogenannte Rückgabewert. Im Gegensatz zu GETDATE() benötigen die meisten Funktionen Eingabeparameter in den nachfolgenden Klammern. Diese Klammern müssen bei Funktionen immer angegeben werden, auch wenn sie keine Parameter enthalten, sonst glaubt der SQL-Server, es handele sich um eine Tabellenspalte, die er nicht findet. Nur wie bekommen wir jetzt den Rückgabewert von GETDATE() auf den Monitor? (Bei Kirk hört sich das immer so einfach an: »Auf den Schirm!«). In SQL gibt es kein System.out.println oder printf. Sie können der Datenbank nicht befehlen, etwas irgendwohin aus*zugeben*. Sie dürfen lediglich aus*wählen*, was zufällig die deutsche Bedeutung des Schlüsselwortes SELECT ist.

Dann mal los:

```
SELECT getdate()
```

Antwort des SQL-Servers:

| | (Kein Spaltenname) |
|---|---|
| 1 | 2012-05-27 18:40:22.507 |

**Abb. 1.1:** Ausgabe von getdate()

Wie man sieht, ist der SQL-Server kein Unmensch. Es ist doch klar, dass wir das Ergebnis sehen wollen. Dabei haben wir das »FROM irgendwoher« in der Abfrage weggelassen, also gar nicht angegeben, wo das gewünschte Ergebnis zu finden ist. Da wir das Ergebnis eines Funktionsaufrufs haben wollen, brauchen wir nicht den Umweg über eine Tabelle zu gehen. Ein SELECT <das_was_ich_haben_will> reicht völlig aus. Es soll noch andere Datenbanken geben, die für solche Zwecke eine Dummy-Tabelle bereithalten müssen, aber das ist sicher nur ein Gerücht ...

Hier gibt es aber noch mehr zu holen als nur das Datum. Haben Sie Durst?

```
SELECT 'Coke'
```

| | (Kein Spaltenname) |
|---|---|
| 1 | Coke |

**Abb. 1.2:** SQL-Server als Getränkeautomat?

### 1.2.3 Hello World

Da ist sie endlich – die Mutter aller Abfragen:

```
SELECT 'Hello World'
```

| | (Kein Spaltenname) |
|---|---|
| 1 | Hello World |

**Abb. 1.3:** Selber Hallo!

Damit wir später noch wissen, was die Antwort bedeutet, können wir die Spalte mit einer Überschrift versehen:

```
SELECT 'Hello World' AS Ausgabe
```

| | Ausgabe |
|---|---|
| 1 | Hello World |

**Abb. 1.4:** Nochmal Hallo

Das Keyword AS ist optional, aber verbessert die Lesbarkeit vor allem der komplizierteren Statements. Die Spaltenbezeichnung, auch »Aliasname« genannt, darf keine Leer- und Sonderzeichen enthalten, es sei denn, man schließt sie in Anführungszeichen oder Hochkommata ein. (Die (doppelten) Anführungszeichen entsprechen dem SQL-Standard und sollten daher vorgezogen werden.)

```
SELECT 2+2 AS "Das Ergebnis von 2+2 ist:"
```

| | Das Ergebnis von 2+2 ist: |
|---|---|
| 1 | 4 |

**Abb. 1.5:** SQL als Taschenrechner

Eine Frage, eine Antwort. So einfach ist SQL. Jedenfalls anfangs ...

Wenn Sie mal wieder unter Zeitdruck eine Budgetübersicht erstellen müssen, hier eine kleine Hilfestellung:

```
SELECT 'Budgetübersicht'
```

## 1.3 Abfrage von echten Tabellen

Wir behalten die Frage bei und stürzen uns damit auf unser erstes Opfer, die Tabelle `commerce.customers`.

```
SELECT 2+2 FROM commerce.customers
```

| | (Kein Spaltenname) |
|---|---|
| 1 | 4 |
| 2 | 4 |
| 3 | 4 |
| 4 | 4 |
| 5 | 4 |
| 6 | 4 |
| 7 | 4 |
| 8 | 4 |
| 9 | 4 |
| 10 | 4 |
| 11 | 4 |
| 12 | 4 |

**Abb. 1.6:** Einmal hätte gereicht

Zwölf Mal die gleiche Antwort ist irgendwie sinnlos. Gehen wir der Sache auf den Grund.

### 1.3.1 Anzeige von Tabelleninhalten – und mehr

Wenn wir Informationen aus einer Tabelle holen wollen, die tatsächlich drinstehen, müssen wir im SELECT-Statement die entsprechenden Spalten angeben. Kennt man den Aufbau der Tabelle noch nicht, kann man sich diesen im Objekt-Explorer (Navigationsfenster auf der linken Bildschirmseite) anzeigen lassen.

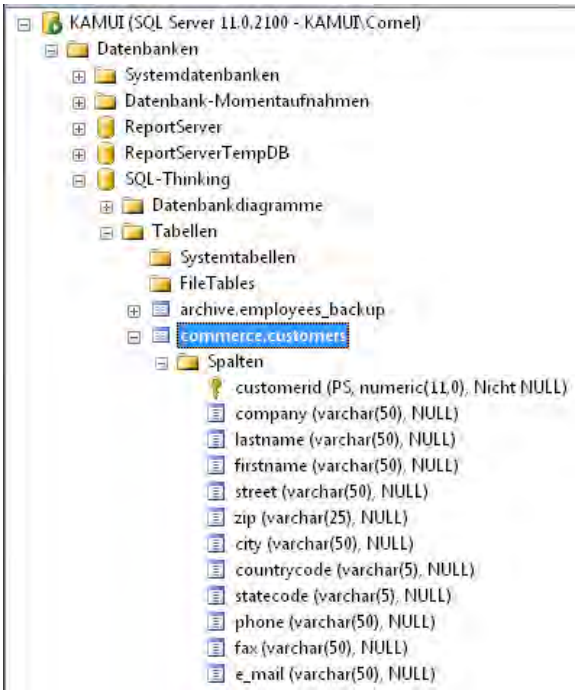


**Abb. 1.7:** Tabellenspalten im Objekt-Explorer

Durch Rechtsklick auf die Tabelle `commerce.customers` öffnet sich das Kontextmenü, in dem wir durch Anklicken von *Oberste 1000 Zeilen auswählen* ein neues Abfragefenster öffnen, das ein SELECT-Statement für die ersten 1000 Zeilen enthält und sofort ausführt.

```
/****** Skript für SelectTopNRows-Befehl aus SSMS ******/
SELECT TOP 1000 [customerid]
      ,[company]
      ,[lastname]
      ,[firstname]
      ,[street]
      ,[zip]
      ,[city]
      ,[countrycode]
      ,[statecode]
      ,[phone]
      ,[fax]
      ,[e_mail]
  FROM [SQL-Thinking].[commerce].[customers]
```

**Listing 1.1:** Das generierte Statement

| | customerid | company | lastname | firstname | street | zip | city | coun |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | Miller Ltd. | Miller | Sean | Elm Street | WC2E 7LB | London | GB |
| 2 | 2 | Müller GmbH | Müller | Siegfried | Bahnhofstr. 12 | 60311 | Frankfurt | DE |
| 3 | 3 | Cordonbleu SA | Cordonbleu | Pierre | Rue de la Concorde 12 | 75001 | Paris | FR |
| 4 | 4 | Meier AG | Meier | Michael | Hauptstr. 23 | 64283 | Darmstadt | DE |
| 5 | 5 | Schmidt OHG | Schmidt | Willi | Postweg 17 | 20095 | Hamburg | DE |
| 6 | 6 | Schulze GmbH | Schulze | Max | Feldbergstr. 77 | 79098 | Freiburg | DE |
| 7 | 7 | Camembert SA | Camembert | Jaques | Avenue Fromage 3 | 13011 | Marseille | FR |
| 8 | 8 | Foster plc | Foster | Henry | Main Street | EH14 2SP | Edinburgh | GB |
| 9 | 9 | Zocker AG | Zocker | Karl | Zuse-Str. 83 | 80331 | München | DE |
| 10 | 10 | Baller GmbH | Baller | Toni | Malle-Weg 93 | 63065 | Offenbach | DE |
| 11 | 11 | Cambozola s.r.l. | Toni | Cambozola | Via Roma 98 | 20100 | Milano | IT |
| 12 | 12 | Cheddar plc | Cheddar | Charles | Cracker Street 5 | BN2 0GN | Brighton | GB |

**Abb. 1.8:** Inhalt der Tabelle commerce.customers

Ja, die Klick-Lösung ist sehr bequem, aber wir konzentrieren uns wieder auf das SQL-Fenster. Wenn Microsoft wollte, dass man diese Datenbank nur mit der Maus bedient, hieße sie Mouse Server 2012 und nicht SQL Server 2012.

Für die handgeschriebene Lösung gibt es auch eine Abkürzung. Ein Sternchen (*) für alle Spalten spart Schreibarbeit:

```
SELECT * FROM commerce.customers
```

Das Ergebnis ist das Gleiche. Die Klausel »TOP 1000«, mit der wir die Ausgabe auf maximal 1000 Zeilen beschränken, ist bei dieser Tabelle mit insgesamt 12 Zeilen überflüssig. Sie sollten sich das aber schon mal merken, denn für große Tabellen ist es echt praktisch, die Ausgabe auf die ersten Zeilen beschränken zu können.

### Tipp

Die Klausel TOP *n* beschränkt die Ausgabe eines SELECT-Statements auf die ersten *n* Zeilen.

Von 2+2=4 ist hier nichts mehr zu sehen, aber die Anzahl der Records sollte Ihnen schon irgendwie bekannt vorkommen. Wir haben bei dem »2+2«-SELECT für jede Zeile in `customers` ein Ergebnis enthalten. Was hat das mit dem Inhalt von `customers` zu tun? Nichts! Verbinden wir zum Vergleich beide Abfragen und hängen zusätzlich die Funktion GETDATE() dran:

```
SELECT company
,      lastname
,      firstname
,      2+2
,      GETDATE()
FROM commerce.customers
```

Ergebnis:

| | company | lastname | firstname | (Kein Spaltenname) | (Kein Spaltenname) |
|---|---|---|---|---|---|
| 1 | Miller Ltd. | Miller | Sean | 4 | 2012-06-03 18:22:56.290 |
| 2 | Müller GmbH | Müller | Siegfried | 4 | 2012-06-03 18:22:56.290 |
| 3 | Cordonbleu SA | Cordonbleu | Pierre | 4 | 2012-06-03 18:22:56.290 |
| 4 | Meier AG | Meier | Michael | 4 | 2012-06-03 18:22:56.290 |
| 5 | Schmidt OHG | Schmidt | Willi | 4 | 2012-06-03 18:22:56.290 |
| 6 | Schulze GmbH | Schulze | Max | 4 | 2012-06-03 18:22:56.290 |
| 7 | Camembert SA | Camembert | Jaques | 4 | 2012-06-03 18:22:56.290 |
| 8 | Foster plc | Foster | Henry | 4 | 2012-06-03 18:22:56.290 |
| 9 | Zocker AG | Zocker | Karl | 4 | 2012-06-03 18:22:56.290 |
| 10 | Baller GmbH | Baller | Toni | 4 | 2012-06-03 18:22:56.290 |
| 11 | Cambozola s.r.l. | Toni | Cambozola | 4 | 2012-06-03 18:22:56.290 |
| 12 | Cheddar plc | Cheddar | Charles | 4 | 2012-06-03 18:22:56.290 |

**Abb. 1.9:** Customers und Sonstiges

Für jede Zeile in der Tabelle werden die gewünschten Spalten ausgegeben. Die Spalten »2+2« und »GETDATE()« sind inhaltlich völlig unabhängig von der abgefragten Tabelle `customers`.

**Wichtig**

Man unterscheidet zwischen Tabellenspalten und Ergebnisspalten. Eine Ergebnisspalte eines SELECT-Statements muss nicht zwingend einen Zusammenhang mit dem Inhalt der ausgewählten Tabelle(n) haben. Sie kann das Ergebnis von beliebigen Berechnungen und Funktionen enthalten.

### 1.3.2 Die erste Textaufgabe

Wir können nicht nur, wie oben vorgeführt, zusätzliche Spalten ausgeben, sondern auch die Feldinhalte der Tabelle verändert ausgeben. Da bietet sich eine Währungsumrechnung an, und so stellen wir als nächstes Beispiel folgende Frage:

Wie hoch ist das Gehalt der Mitarbeiter in Euro?

An diese Frage wollen wir schon etwas methodischer herangehen. Erinnern wir uns nochmal an den prinzipiellen Aufbau eines SELECT-Statements: »SELECT *irgendwas* FROM *irgendwoher*«. Wir sollten mit dem *irgendwoher* beginnen, denn wir können erst dann herausfinden, wie die einzelnen Tabellenspalten heißen, wenn wir die benötigten Tabellen identifiziert haben.

1. Schritt: Datenquellen zusammensuchen

Die Tabelle(n) identifizieren, in denen die gesuchten Daten zu finden sind. Die Thematik `hr` (Human Resources) scheint da am besten zu passen und enthält eine Tabelle `employees`.

Eine Tabelle für Währungskurse haben wir nicht, da soll uns eine Konstante genügen: 1$ = 0,804 €

2. Schritt: Die erforderlichen Spalten auswählen

Im richtigen Leben besteht dieser Schritt mindestens aus mehreren Meetings, Abstimmungsprozessen, einem Fachkonzept und einem DV-Konzept. Wir verlassen uns lieber auf den gesunden Menschenverstand und kürzen das hier ab:

| Ergebnisspalte | Tabellenspalte |
|---|---|
| Personalnummer | employee_id |
| Name | last_name |
| Stellenbezeichnung | job_id |
| Gehalt in $ | salary |
| Gehalt in € | salary * 0,804 |

3. Schritt: SELECT-Statement zusammenbauen und testen

```
SELECT employee_id    AS "Personalnummer"
,      last_name              AS "Name"
,      job_id         AS "Stellenbezeichnung"
,      salary         AS "Gehalt $"
,      salary * 0.804 AS "Gehalt €"
FROM   hr.employees
```

Ergebnis:

| | Personalnummer | Name | Stellenbezeichnung | Gehalt $ | Gehalt € |
|---|---|---|---|---|---|
| 1 | 100 | King | AD_PRES | 24000.00 | 19296.00000 |
| 2 | 101 | Kochhar | AD_VP | 17000.00 | 13668.00000 |
| 3 | 102 | De Haan | AD_VP | 17000.00 | 13668.00000 |
| 4 | 103 | Hunold | IT_PROG | 9000.00 | 7236.00000 |
| 5 | 104 | Ernst | IT_PROG | 6000.00 | 4824.00000 |
| 6 | 105 | Austin | IT_PROG | 4800.00 | 3859.20000 |
| 7 | 106 | Pataballa | IT_PROG | 4800.00 | 3859.20000 |
| 8 | 107 | Lorentz | IT_PROG | 4200.00 | 3376.80000 |
| 9 | 108 | Greenberg | FI_MGR | 12008.00 | 9654.43200 |
| 10 | 109 | Faviet | FI_ACCOUNT | 9000.00 | 7236.00000 |
| 11 | 110 | Chen | FI_ACCOUNT | 8200.00 | 6592.80000 |
| 12 | 111 | Sciarra | FI_ACCOUNT | 7700.00 | 6190.80000 |

**Abb. 1.10:** Präzise Frage, präzise Antwort

Es ist unerheblich, ob ein fertiges SELECT-Statement nur einmal verwendet oder in ein Programm oder Reporting-System eingebettet wird. Je mehr Sie auf SQL-

Ebene in der Datenbank leisten, desto weniger manuelle Nacharbeit bleibt außerhalb der Datenbank.

Eine echte Fachabteilung ist natürlich nie zufrieden. Ihre fertige Auswertung hat auf die Zahlenakrobaten die gleiche Wirkung wie Blut auf Vampire. Sie erzeugt einen unstillbaren Hunger nach immer mehr Kennzahlen, in immer neuen Kombinationen. Wir wünschen viel Spaß beim Füttern ;-).