



Cornel
Brücher
Frank
Jüdes
Wulf
Kollmann

SQL Thinking

Vom Problem zum SQL-Statement



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 SQL – Die Sprache des Orakels

Die Antworten des Orakels aus Redwood sind zwar verständlicher als damals bei den alten Griechen in Delphi, dafür ist die Befragung etwas komplizierter. Dieses Orakel versteht nur SQL (wenn Sie das »Sssickwl« aussprechen, outen Sie sich als Experte. Deswegen heißt dieses Buch auch »Sssickwl Thinking«).

SQL (Structured Query Language) ist die standardisierte Datenzugriffssprache für relationale Datenbanken. SQL ist keine Programmiersprache, da sich keine Programmabläufe, Verzweigungen oder Schleifen formulieren lassen. 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). 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. Sie werden sehen, dass man mit SQL-Abfragen auf einer Oracle-Datenbank Auswertungen erzeugen kann, die normalerweise mit viel Nach(t)arbeit und einem schwarzen Gürtel in Excel erledigt werden.

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. 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 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 Seiteneffekt 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 (betriebswirtschaftlichen) 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

In diesem Kapitel arbeiten wir mit dem User **SCOTT**, der zu jeder Oracle-Datenbank mitgeliefert wird und in seinem »Schema« ein stark vereinfachtes Datenmodell aus dem Bereich Human Resources bereithält. Dieses ist aber aus Sicherheitsgründen noch gesperrt. Zur Freischaltung sind die folgenden Schritte notwendig:

1. Melden Sie sich als **SYSTEM** an.
2. Entsperren Sie den User mit: **ALTER USER scott ACCOUNT UNLOCK.**
3. Setzen Sie das Passwort mit: **ALTER USER Scott IDENTIFIED BY qwertzuiop.**
4. Melden Sie sich an als **SCOTT**.

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 **SYSDATE**. 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 **SYSDATE** benötigen die meisten Funktionen Parameter. Nur wie kommen wir jetzt an den Rückgabewert heran?

In SQL gibt es kein `System.out.println` oder `printf`. Sie können der Datenbank nicht befehlen, etwas irgendwohin auszugeben. Sie dürfen auswählen. Bitte melden Sie sich an als User **SCOTT**.

Dann mal los:

```
SELECT sysdate
```

Antwort von Oracle:

```
SQL Error: ORA-00923: Schlüsselwort FROM nicht an erwarteter Stelle gefunden
```

Wir haben schon angegeben, was wir haben wollen, aber noch nicht, wo es zu finden ist. Oracle erwartet die Angabe mindestens einer Tabelle, in der das Gesuchte zu finden ist, hinter dem Schlüsselwort **FROM**. Hier kommt die Tabelle **DUAL** ins Spiel. In jeder Oracle-Datenbank existiert im Schema des Administrator-Users **SYS** die Tabelle **DUAL** mit genau einer Spalte und genau einer Zeile. Diese Tabelle wird benötigt, um Abfragen durchzuführen, die nur eine Ergebniszeile liefern sollen, und mit dem Inhalt irgendwelcher Tabellen nichts zu tun haben. Das können Berechnungen sein oder wie in diesem Beispiel der Aufruf von SQL-Funktionen.

```
SELECT sysdate FROM dual
```

Kapitel 1

Das Problem mit dem Problem

Antwort vom Orakel:

		SYSDATE	
▶	1	22.05.2010 15:33:34	▼

Abb. 1.1: Des Orakels erste Antwort

DUAL ist so eine Art Tante-Emma-Laden in der Oracle-Datenbank. Da gibt's einfach alles. Durst?

```
SELECT 'Coke' FROM dual
```

Ergebnis:

		'COKE'	
▶	1	Coke	

Abb. 1.2: Ein Getränkeautomat?

1.2.3 Hello World

Dann kommen wir mal zur Mutter aller Abfragen, und erleben das »Hello world«-Feeling von SQL:

```
SELECT 'Hello world' FROM dual
```

Ergebnis:

		'HELLOWORLD'	
▶	1	Hello World	

Abb. 1.3: Selber Hallo!

Oracle kann es nicht lassen, auch diese einfachen Ergebnisse mit einer Spaltenüberschrift zu versehen. Dann wollen wir uns diese wenigstens aussuchen:

```
SELECT 'Hello World' AS Ausgabe FROM dual
```

Ergebnis:

		AUSGABE	
▶	1	Hello World	

Abb. 1.4: Nochmal Hallo

Schon besser. Das Keyword **AS** ist optional, aber verbessert die Lesbarkeit vor allem der komplizierteren Statements. Der Aliasname wird im Ergebnis immer in

Großbuchstaben ausgegeben und darf keine Leer- und Sonderzeichen enthalten, es sei denn, man schließt ihn in Anführungszeichen ein.

```
SELECT 2+2 AS "Mein Ergebnis. Toll, wa?" FROM dual
```

Ergebnis:

	Mein Ergebnis. Toll, wa?
▶ 1	4

Abb. 1.5: SQL-Taschenrechner

Faszinierend, was diese Tabelle so alles weiß. 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' FROM dual
```

1.3 Abfrage von echten Tabellen

Dual ist zwar hin und wieder praktisch, aber nicht die einzige Tabelle in unserer Datenbank. Suchen wir uns ein neues Opfer. Der User SCOTT, unter dem wir uns angemeldet haben, besitzt mehrere Tabellen. Eine davon ist EMP. Mal sehen, ob die auch so schlau ist wie dual:

```
SELECT 2+2 FROM emp
```

Ergebnis:

	2+2
▶ 1	4
2	4
3	4
4	4
5	4
6	4
7	4
8	4
9	4
10	4
11	4
12	4
13	4
14	4

Abb. 1.6: Zuviel des Guten

Vierzehn mal so schlau. Eine Antwort hätte auch gereicht. Irgendwie war das mit DUAL einfacher. Gehen wir der Sache auf den Grund.

1.3.1 Anzeige von Tabelleninhalten – und mehr

Wenn wir Informationen aus einer Tabelle holen wollen, die zur Abwechslung tatsächlich drinstehen, müssen wir im SELECT-Statement die entsprechenden Spalten angeben. Kennt man den Aufbau der Tabelle noch nicht, kann man sich diesen durch Herumklicken im Object Browser (Navigationsfenster auf der linken Bildschirmseite) anzeigen lassen und hinter dem SELECT brav die einzelnen Spaltenbezeichnungen eintippen. Man muss aber nicht, ein Sternchen (*) für alle Spalten tut's auch.

```
SELECT * FROM emp
```

Ergebnis:

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
▶ 1	7369	SMITH	CLERK	7902	17.12.1980	800,00		20
2	7499	ALLEN	SALESMAN	7698	20.02.1981	1600,00	300,00	30
3	7521	WARD	SALESMAN	7698	22.02.1981	1250,00	500,00	30
4	7566	JONES	MANAGER	7839	02.04.1981	2975,00		20
5	7654	MARTIN	SALESMAN	7698	28.09.1981	1250,00	1400,00	30
6	7698	BLAKE	MANAGER	7839	01.05.1981	2850,00		30
7	7782	CLARK	MANAGER	7839	09.06.1981	2450,00		10
8	7788	SCOTT	ANALYST	7566	19.04.1987	3000,00		20
9	7839	KING	PRESIDENT		17.11.1981	5000,00		10
10	7844	TURNER	SALESMAN	7698	08.09.1981	1500,00	0,00	30
11	7876	ADAMS	CLERK	7788	23.05.1987	1100,00		20
12	7900	JAMES	CLERK	7698	03.12.1981	950,00		30
13	7902	FORD	ANALYST	7566	03.12.1981	3000,00		20
14	7934	MILLER	CLERK	7782	23.01.1982	1300,00		10

Abb. 1.7: Die gesamte Tabelle EMP

Von 2+2=4 ist hier nichts zu sehen, aber die Anzahl der Records sollte Ihnen schon irgendwie bekannt vorkommen. Wir haben bei dem »2+2«-SELECT für jede Zeile in EMP ein Ergebnis enthalten. Was hat das mit dem Inhalt von EMP zu tun? Nichts!

Verbinden wir zum Vergleich beide Abfragen und hängen zusätzlich die Funktion sysdate dran:

```
SELECT empno, ename, job, sal, 2+2, sysdate
FROM emp
```

Ergebnis:

	EMPNO	ENAME	JOB	SAL	2+2	SYSDATE
▶ 1	7369	SMITH	CLERK	800,00	4	28.12.2009 ▼
2	7499	ALLEN	SALESMAN	1600,00	4	28.12.2009 ▼
3	7521	WARD	SALESMAN	1250,00	4	28.12.2009 ▼
4	7566	JONES	MANAGER	2975,00	4	28.12.2009 ▼
5	7654	MARTIN	SALESMAN	1250,00	4	28.12.2009 ▼
6	7698	BLAKE	MANAGER	2850,00	4	28.12.2009 ▼
7	7782	CLARK	MANAGER	2450,00	4	28.12.2009 ▼
8	7788	SCOTT	ANALYST	3000,00	4	28.12.2009 ▼
9	7839	KING	PRESIDENT	5000,00	4	28.12.2009 ▼
10	7844	TURNER	SALESMAN	1500,00	4	28.12.2009 ▼
11	7876	ADAMS	CLERK	1100,00	4	28.12.2009 ▼
12	7900	JAMES	CLERK	950,00	4	28.12.2009 ▼
13	7902	FORD	ANALYST	3000,00	4	28.12.2009 ▼
14	7934	MILLER	CLERK	1300,00	4	28.12.2009 ▼

Abb. 1.8: EMPs und Sonstiges

Für jede Zeile in der Tabelle werden die gewünschten Spalten ausgegeben. Die Spalten »2+2« und »SYSDATE« sind inhaltlich völlig unabhängig von der abgefragten Tabelle EMP.

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. Die Spalte SAL enthält offensichtlich das als Salary bezeichnete Gehalt und wird sich wohl auf Dollar beziehen. Da wir in Euro rechnen, stellen wir als nächstes Beispiel folgende Frage an das Orakel:

Wie hoch ist das Gehalt der Mitarbeiter in Euro?

An diese Frage wollen wir schon etwas methodischer herangehen. Der aktuelle Dollarkurs steht nicht in unserer Datenbank, also nehmen wir der Einfachheit halber eine Konstante: 0,80 €.

Die prinzipielle Struktur eines SELECT-Statements ist einfach:

```
SELECT spalte_1, spalte_2, ... , spalte_n
FROM   tabelle
```

Wie gesagt: »SELECT *irgendwas* VON *irgendwoher*«. Wir empfehlen Ihnen, mit dem »irgendwoher« zu beginnen, denn Sie können erst dann herausfinden, wie die einzelnen Tabellenspalten heißen, wenn Sie zuvor die Tabellen identifiziert haben.

Dann wollen wir mal:

1. Schritt: Datenquellen zusammensuchen

Die Tabelle(n) identifizieren, in denen die gesuchten Daten zu finden sind. Wir wissen schon, dass alle benötigten Felder mit Ausnahme des zu berechnenden Euro-Betrags sowie des Dollarkurses in der Tabelle EMP zu finden sind.

Das Gehalt in Euro wird eine berechnete Ergebnisspalte sein, und der Dollarkurs wurde als Konstante (0,80 €) festgelegt.

2. Schritt: Die erforderlichen Spalten der Quelltabellen 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	EMPNO
Name	ENAME
Stellenbezeichnung	JOB
Gehalt	SAL

3. Schritt: SELECT-Statement zusammenbauen und testen

```
SELECT empno
,      ename
,      job
,      sal
,      sal * 0.8
FROM   emp
```

Wichtig

Die Schritte vom Problem zum SQL-Statement kurz zusammengefasst:

1. Die benötigten Tabellen ermitteln
2. Die erforderlichen Spalten auswählen
3. SELECT-Statement zusammenbauen und testen

Ergebnis:

	EMPNO	ENAME	JOB	SAL	SAL*0.8
▶ 1	7369	SMITH	CLERK	800,00	640
2	7499	ALLEN	SALESMAN	1600,00	1280
3	7521	WARD	SALESMAN	1250,00	1000
4	7566	JONES	MANAGER	2975,00	2380
5	7654	MARTIN	SALESMAN	1250,00	1000
6	7698	BLAKE	MANAGER	2850,00	2280
7	7782	CLARK	MANAGER	2450,00	1960
8	7788	SCOTT	ANALYST	3000,00	2400
9	7839	KING	PRESIDENT	5000,00	4000
10	7844	TURNER	SALESMAN	1500,00	1200
11	7876	ADAMS	CLERK	1100,00	880
12	7900	JAMES	CLERK	950,00	760
13	7902	FORD	ANALYST	3000,00	2400
14	7934	MILLER	CLERK	1300,00	1040

Abb. 1.9: Präzise Frage, präzise Antwort

Es ist unerheblich, ob ein fertiges SELECT-Statement in ein Programm oder Reporting-System eingebettet wird. Je mehr Sie auf SQL-Ebene in der Datenbank leisten, umso 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 ;-).