Zahlen zum Malen

Wie aus Produktionszahlen - unter Einsatz von rekursivem SQL der IBM DB2 und der PHP-Bibliothek JpGraph - ansprechende Auswertungen werden können.

von Thomas Wiedmann

Gesetzt den Fall, der Fertigungsleiter kommt mit gehetztem Gesichtsausdruck zu Ihnen gerannt und fragt, ob Sie nicht schnell mal herausfinden könnten, ob die Produktionszahlen über die Sommerferien noch dem Soll entsprechend und das Ganze als Grafik für den Chef aufbereiten können. Was tun Sie? a) Zuständigkeit prüfen? , b) Unsichtbar werden? oder c) sich entspannt zurück lehnen und diesen Artikel lesen? Wie immer Sie sich auch entscheiden, Grundlage für dieses Tutorial hier sind erst einmal folgende Komponenten:

- Windows 2000
- Apache 1.3.8
- DB2/NT 8.1
- PHP 5.0.1
- JpGraph 2.0Alpha [1]

Damit Sie sehen, auf was Sie sich hier einlassen, zeige ich Ihnen hier das Ergebnis, mit dem Sie den hektischen Fertigungsleiter beruhigen und den kritischen Chef überzeugen können.

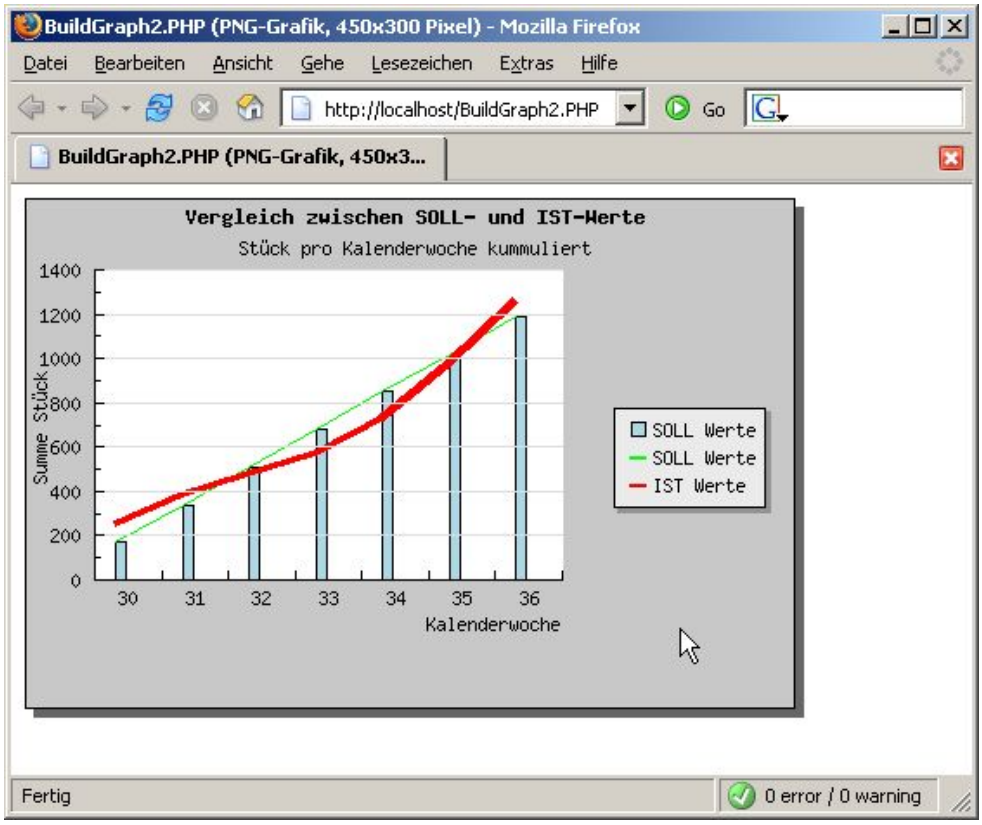


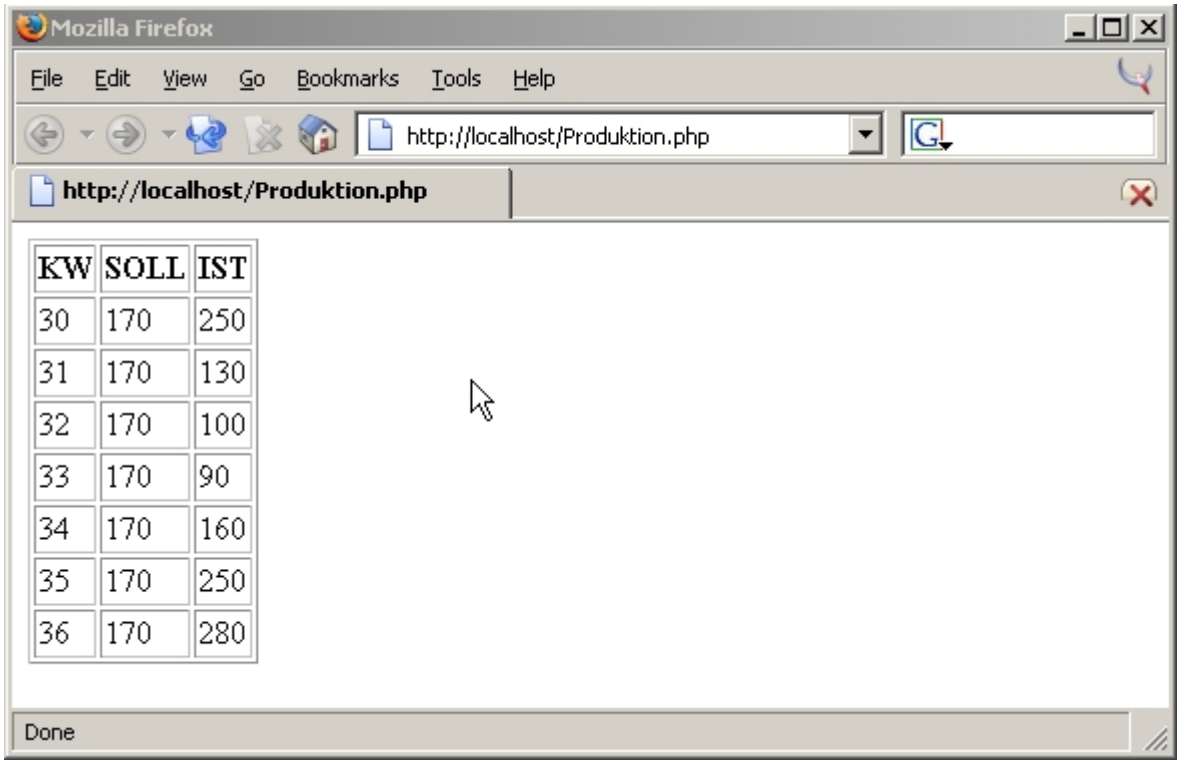
Abb. 1: Zeit eine Gegenüberstellung der SOLL- und IST-Werte aus der Produktion (Powered by JpGraph)

Ziel ist es also, die einzelnen Produktionszahlen (IST-Werte Grün) den geplanten (SOLL-Werten Rot) gegenüberzustellen. Der Clou dabei ist, es werden nicht nur die Wochenwerte pro Kalenderwoche (KW) dargestellt, sondern es wird ein festgelegter Zeitraum (KW 30 – 36) betrachtet. Hierbei entstehen kumulierte Summen. Nur so ist es möglich zu erkennen, ob die

Produktion die Sollzahlen – trotz Sommerferien – einhalten und gegebenenfalls wieder aufholen konnte.

Die Datenbasis

Grundlage der Überlegungen ist natürlich das Wissen um die vorhandene Datenbasis. Die Produktionszahlen (IST-Werte) sowie die geplanten Zahlen (SOLL-Werte) landen in der Tabelle PRODUKTION (Abb. 2).



KW	SOLL	IST
30	170	250
31	170	130
32	170	100
33	170	90
34	170	160
35	170	250
36	170	280

Abb. 2: Die Datenbasis für den Vergleich der SOLL- und IST-Zahlen

Pro Kalenderwoche sind in der Tabelle PRODUKTION die SOLL- und IST-Werte gespeichert. Die Aufgabe ist es nun, aus diesem Zahlenmaterial eine geeignete grafische Gegenüberstellung zu erzeugen, inwieweit Plan und Wirklichkeit übereinstimmen. Es genügt also nicht jede Woche für sich zu Betrachen, sondern es muss geprüft werden, ob innerhalb des gewählten Zeitraumes (KW 30-36) die geforderten Zahlen - in Summe betrachtet - erreicht werden. Nur so ist es möglich, einen Produktionsengpass zu erkennen und gegebenenfalls Gegenmaßnahmen einzuleiten.

```
SQL #1:

SELECT SUM(soll) AS sum_soll,
       SUM(ist) AS sum_ist
  FROM tw.produktion;

   SUM_SOLL  SUM_IST
========== ==========
       1190     1260
```

Mit SQL #1 lässt sich zwar schnell ermitteln, ob die Produktion auf dem Laufenden ist, aber es ist nicht zu erkennen, in welcher KW der Plan erfüllt worden ist.

```
SQL #2:

SELECT kw,
       SUM(soll) AS sum_soll,
       SUM(ist) AS sum_ist,
```

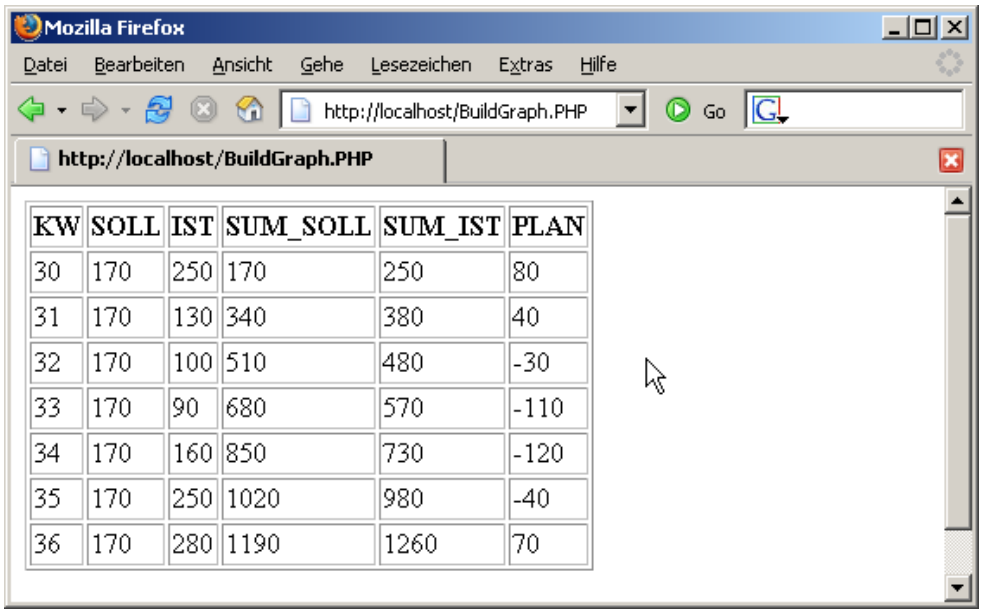
```
       SUM(ist) - SUM(soll)
        AS zw_sum
  FROM tw.produktion
 GROUP BY kw
 ORDER BY kw;

KW    SUM_SOLL   SUM_IST  ZW_SUM
===== ========== ======== =======
 30     170        250      80
 31     170        130     -40
 32     170        100     -70
 33     170         90     -80
 34     170        160     -10
 35     170        250      80
 36     170        280     110
```

Durch einen GROUP BY über die Spalte Feld KW (SQL #2) besteht die Möglichkeit, die Produktionsabweichung (ZW_SUM) pro KW sichtbar zu machen. Allerdings genügt es dem Produktionsleiter nicht, nur die Endsummen oder Zwischenergebnisse zu betrachen. Was wir brauchen, sind Zahlen, die für jede KW den erreichten Stand gemäß dem Plan zeigen. Wann wurde zum Beispiel der Rückstand aus KW 32 wieder aufgeholt?

SOLL- / IST-Vergleich für Fortgeschrittene

Vermutlich denken Sie jetzt an eine Art Schleife, die die Tabelle PRODUKTION einliest und den jeweils erreichten Stand von SOLL und IST, als Summe ausgibt. Das Ergebnis könnte so aussehen (Abb. 3).



KW	SOLL	IST	SUM_SOLL	SUM_IST	PLAN
30	170	250	170	250	80
31	170	130	340	380	40
32	170	100	510	480	-30
33	170	90	680	570	-110
34	170	160	850	730	-120
35	170	250	1020	980	-40
36	170	280	1190	1260	70

Abb. 3: Zeigt die Einzel-, Summen als auch den erreichten Plan

Und richtig, was Sie hier sehen, ist die Grundlage für die eingangs gezeigte grafische Auswertung (Abb. 1). Nun wird sichtbar, dass es im Zeitraum KW 32 bis KW 35 einen Produktionsrückstand gab, obwohl zum Teil die IST-Zahlen deutlich über den SOLL-Werten lagen. Aber glauben Sie mir auch, wenn ich jetzt noch behaupte, dass die Zahlen für die Abbildung 3 mit einem einzigen SQL aufbereitet werden? Das Stichwort dafür ist »Rekursion« beziehungsweise »Rekursives SQL«.

Rekursives was ..? – Rekursives SQL!

Zitat: *Als Rekursion bezeichnet man den Aufruf oder die Definition einer Funktion durch sich selbst* [2]. In die SQL-Welt übersetzt bedeutet dies, eine Abfrage, die ihr eigenes Ergebnis

nutzt. Wohl dem, der da mit DB2 (WITH) oder ORACLE (CONNECT BY) arbeitet. Alle anderen müssen diese Funktionalität im Programmcode oder als Stored Procedure nachbilden. Wie PHP und DB2 Stored Procedures zusammenarbeiten, habe ich bereits im PHP Magazin beschrieben [3]. Doch ich will Sie nun nicht weiter auf die Folter spannen, in Listing 1 sehen Sie rekursives SQL ala IBM DB2.

Listing 1:

BuildGraph.php: (Ausschnitt)

```
[...]
/***
* DB2 Rekursives SQL ausführen
*/
$sSQL = "";

// (1)
$sSQL = $sSQL."WITH temp_prod ";
$sSQL = $sSQL."     (kw, ";
$sSQL = $sSQL."      soll, ";
$sSQL = $sSQL."      ist, ";
$sSQL = $sSQL."      sum_soll, ";
$sSQL = $sSQL."      sum_ist, ";
$sSQL = $sSQL."      kwp1) AS ";
$sSQL = $sSQL."( ";

// (2)
$sSQL = $sSQL." SELECT kw, ";
$sSQL = $sSQL."        soll, ";
$sSQL = $sSQL."        ist, ";
$sSQL = $sSQL."        soll, ";
$sSQL = $sSQL."        ist, ";
$sSQL = $sSQL."        kw +1 ";
$sSQL = $sSQL."   FROM tw.produktion ";
$sSQL = $sSQL."  WHERE kw = 30 ";
$sSQL = $sSQL."  ";

// (3)
$sSQL = $sSQL." UNION ALL ";
$sSQL = $sSQL."  ";
$sSQL = $sSQL." SELECT p.kw, ";
$sSQL = $sSQL."         p.soll, ";
$sSQL = $sSQL."         p.ist, ";
$sSQL = $sSQL."         p.soll + tp.sum_soll, ";
$sSQL = $sSQL."         p.ist + tp.sum_ist, ";
$sSQL = $sSQL."         p.kw +1 ";
$sSQL = $sSQL."    FROM tw.produktion p, ";
$sSQL = $sSQL."         temp_prod tp ";
$sSQL = $sSQL."   WHERE p.kw = tp.kwp1 ";
$sSQL = $sSQL.") ";

// (4)
$sSQL = $sSQL." SELECT kw, ";
$sSQL = $sSQL."        soll, ";
$sSQL = $sSQL."        ist, ";
$sSQL = $sSQL."        sum_soll, ";
$sSQL = $sSQL."        sum_ist ";
$sSQL = $sSQL."   FROM temp_prod ";
$sSQL = $sSQL."  WHERE kw BETWEEN 30 AND 36 ";
$sSQL = $sSQL." ORDER BY kw";

// SQL ausführen
$hResult = odbc_exec($hDB,$sSQL);
if ($hResult) {
[...]
```

Was hier wie ein Endloswurm aussieht, sind eigentlich vier SQL-Teile, die zusammenarbeiten. Mit WITH (1) werden zwei SQL-Befehle (2)+(3) zu einer Schleife geklammert. Die Befehle (2) + (3) sind mit einen UNION ALL verbunden. Wobei WITH im Prinzip eine temporäre Zwischentabelle TEMP_PROD darstellt, die letztlich von (4) mit einem Standard-SQL abgefragt wird. Besonderes Augenmerk müssen Sie auf (3) werfen. Hier steht der JOIN zwischen der eigentlichen Tabelle PRODUKTION und der temporären

Zwischentabelle TEMP_PROD. Mit (2) startet die Rekursion und endet, wenn (3) keine Daten mehr findet. Sie merken schon, hier ist eine Art Schleife, die die eigenen Ergebnisse weiter verwendet. An dieser Stelle ist auch eventuell Ungemach versteckt, denn unter ungünstigen Voraussetzungen (bei diesem Beispiel nicht) ist hier eine Endlosschleife möglich. Zu weiteren Details zum Thema »Rekursives SQL« möchte ich auf die Literatur beispielsweise [4] oder die DB2 Systemliteratur verweisen. Damit ist die erste Stufe erklommen. Die Daten sind in einer günstigen Form aufbereitet (Abb. 3), um sie mit MS-Excel, JpGraph oder anderen Analyse- und Grafiktools problemlos aufbereiten zu können.

JpGraph in Action

Nachdem uns die Datenaufbereitung ganz locker von der Hand gegangen ist, machen wir uns daran, das Ganze zu visualisieren. Da der Einstieg in JpGraph hier schon ausführlich erläutert worden ist [5], möchte ich gleich mitten rein springen. Zur Installation nur vielleicht so viel, ich habe die JpGraph Zip-Datei in einem eigenen Verzeichnis entpackt und das Startverzeichnis von Apache (httpd.conf , DocumentRoot) auf dieses neue Verzeichnis eingestellt und den »Indianer« neu gestartet. Ach ja, die GD2 Bibliothek muß in der PHP.INI aktiv sein. JpGraph liefert dazu noch ein kleines Testprogramm mit, das ich für Sie aus der Dokumentation extrahiert habe (siehe *test_gd2.php*). Weiterhin muss in der JpGraph Konfiguration *jpg-config.inc* die Konstante TTF_DIR auf die Fonts verweisen: .z.B. DEFINE("TTF_DIR","C:/WINNT/Fonts/"); .

Listing 2:

test_gd2.php

```
<?php

 /**
 * jpgraph-1.16/docs/html/2installation.html#3_1_1
 *
 * 3.1.1 Verifying that you have GD2 installed
 *       To access the more advanced features of JpGraph you need GD 2.x library.
 *       This will allow you to use features as alphablending and trucolor images.
 *       To make sure that you have GD 2.x the following script must be working.
 */
 $im = imagecreatetruecolor (300, 200);
 $black = imagecolorallocate ($im, 0, 0, 0);
 $white = imagecolorallocate ($im, 255, 255, 255);

 imagefilledrectangle($im,0,0,399,99,$white);
 imagerectangle($im,20,20,250,190,$black);

 header ("Content-type: image/png");
 imagepng ($im);
?>
```

Mein Testverzeichnis (DocumentRoot) sieht somit wie folgt aus (Abb. 4).

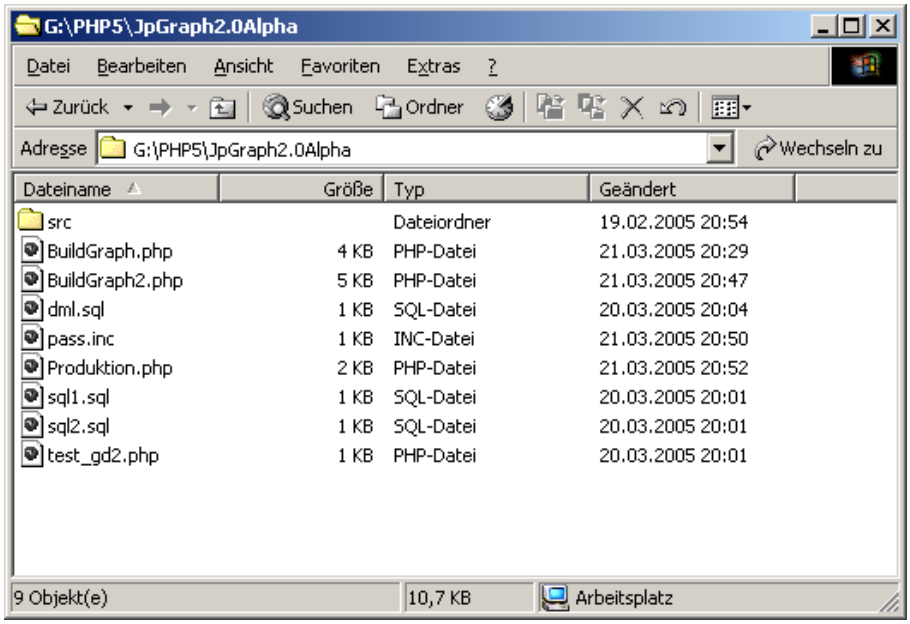


Abb. 4: Inhalt des DocumentRoot Verzeichnisses für diesen Artikel

Im zweiten Schritt bereiten wir die Daten für JpGraph auf. Das bedeutet, die Spalten *KW*, *SumSOLL* und *SumIST* werden in Arrays eingelesen, um anschließend übergeben zu werden.

Listing 3:

BuildGraph2.php: (Ausschnitt)

```
[...]
// SQL ausführen
$hResult = odbc_exec($hDB,$sSQL);
if ($hResult) {

 // anzahl Spalten ermitteln
 $nCol = odbc_num_fields($hResult);

 // Alle Datenzeilen in Arrays einlesen
 while (odbc_fetch_row($hResult)) {

   $aKW[] = odbc_result($hResult,1);
   $aSumSOLL[] = odbc_result($hResult,4);
   $aSumIST[] = odbc_result($hResult,5);
 }
}
[...]
```

Wenn Sie es bis hierher geschafft haben, ist alles weitere einfach und eher ein angenehmes Ausprobieren der sehr gut dokumentierten API und der großen Menge an Beispielen. Einen kurzen Blick in den Quellcode möchte ich Ihnen als Anregung noch erlauben, aber keine Angst, der komplette Source wird zusammen mit diesem Artikel geliefert.

Listing 4:

BuildGraph2.php: (Ausschnitt)

```
[...]
// SOLL-Werte als grüne Linie ausgeben
 $p2 = new LinePlot($aSumSOLL);
 $p2->SetColor("green");
 $p2->SetLegend("SOLL Werte");

 // SOLL-Werte zusätzlich als Balkendiagramm
 $b1 = new BarPlot($aSumSOLL);
```

```
 $b1->SetLegend("SOLL Werte");
 $b1->SetAbsWidth(6);

 // Die Reihenfolge entscheidet, was "on top" ist
 $graph->Add($b1);
 $graph->Add($p2);
 $graph->Add($p1);


 // Image ausgeben
 $graph->Stroke();
[...]
```

Nun möchte ich Sie nicht weiter aufhalten, aber bedenken Sie, zur Zeit ist JpGraph 2.0 Alpha erst zu Testzwecken für PHP5 freigegeben. Zum aktuellen Entwicklungsstand schauen Sie am besten immer wieder mal auf die JpGraph Homepage [1].

Resumee

Der wesentliche Trick, um schnell eine aussagefähige Grafik zu bekommen ist, das Zahlenmaterial der Grafikbibliothek möglichst mundgerecht zu servieren. Dies kann unter Umständen eine ganze Menge Gehirnschmalz verbrauchen, bis der SQL »passt«, aber erspart so manche spektakuläre Programmschleife. DB2 liefert mit dem gezeigten Konzept für rekursives SQL einen wirkungsvollen und mächtigen Ansatz. Nun kann JpGraph aus den aufbereiteten Zahlen wunderschöne und aussagekräftige Bilder malen.

Thomas Wiedmann ist Anwendungs- und Datenbankentwickler für individuelle Softwarelösungen, Autor des Buches »DB2«, sowie IBM Certified DB2 Administrator. Wenn er nicht gerade Musik hört, liest oder fotografiert, beschäftigt er sich mit Datenbanken und Web-Applikationen.

Literatur und Links

[1] JpGraph: http://www.aditus.nu/jpgraph/
[2] WikiPedia: http://de.wikipedia.org/rekursion
[3] Thomas Wiedmann, PHP ruft DB2. Online-Artikel-Archiv (http://www.php-mag.de).
[4] George Baklarz, Bill Wong: DB2 Universal Database v7.1 for UNIX, Linux, Windows, and OS/2. Prentice Hall 2001. ISBN 0-13-091366-9
[5] Florian Lanthaler, Auf einen Blick. PHP-Magazin 04.2002, Seite 68ff.