Dieser Artikel beschreibt die Einführung einer Chart-Komponente im PL/SQL-Umfeld zentral in der Datenbank und geht auf Integrationstests sowie den ersten Prototyp-Report ein.

Zentrale Chart-Erstellung

Steffen Schumann, Sönke Frahne, Dr. Rüdiger Harmel, Dennis Klemme, Michael Meyer, Christian Schmidt und Andriy Terletsky, Berenberg Bank

Im Banken-Umfeld werden grafische Darstellungen alphanumerischer Werte benötigt, um Vorgänge zu erläutern, Analysen durchzuführen, Sachverhalte zu dokumentieren oder auch nur Kundenreports zu untermalen. Zum Einsatz kommt hierfür bei der Berenberg Bank „PL/PDF“ der OraNext-Incorporation. Diese als Oracle-Source installierte Komponente fügt sich gut in die gestellten Anforderungen bezüglich Sicherheit, Schnelligkeit und Stabilität ein.

Das ebenfalls vom Hersteller erhältliche PL/PDF-Chart-Tool ist jedoch den gegenwärtigen Anforderungen an ein einfaches, zeitgemäßes Charting noch nicht gewachsen. Angestrebt wurden die Vereinheitlichung des Chart-Aussehens und die Zentralisierung der Logik dahinter. Die Erzeugung des Charts sollte aus der Datenbank erfolgen.

Eine Arbeitsgruppe suchte Charting-Tools, prüfte und testete sie. Beachtet wurden Marktgewichtung, Entwicklungszukunft oder auch die Philosophie des Produkts.

Ziel war es, ein Abbild eines Charts (möglichst PNG) aus der Datenbank zu erzeugen, das dann dort weiterverarbeitet werden kann. Vier Komponenten / Tools kamen in die engere Auswahl (jFreeChart, Oracle Chart Builder, MS DevExpress und PL/PDF Chart). Abgewogen wurden Layout, Ausgereiftheit, Technologie, Chart-Typen, Zukunftsaussichten, Verbreitung, Schnittstellen, Support, Kosten und Performance (siehe Tabelle 1):

- JFreeChart stellt eine Vielzahl von Chart-Typen bereit, der Support über die Community ist sehr gut und die Basiskomponente ist kostenlos.
- Oracle Chart Builder wird seit 2002 nicht mehr weiterentwickelt; das Layout der Chart-Typen wirkt altbacken.
- MS DevExpress hat ein modernes Layout und lässt sich mit .NET flexibel anpassen.
- PL/PDF Chart stellt kein komplettes Chart zur Verfügung, sondern nur die Prozeduren zum Zeichnen von Linien/Flächen. Damit wäre eine aufwändige Entwicklung einer Mittelschicht notwendig.

Die Architektur

Eine der wichtigsten und damit ersten Fragen bei MS DevExpress ist die nach der Schnittstellentechnologie zwischen der Datenbank und der .NET-Komponente (siehe Abbildung 1).

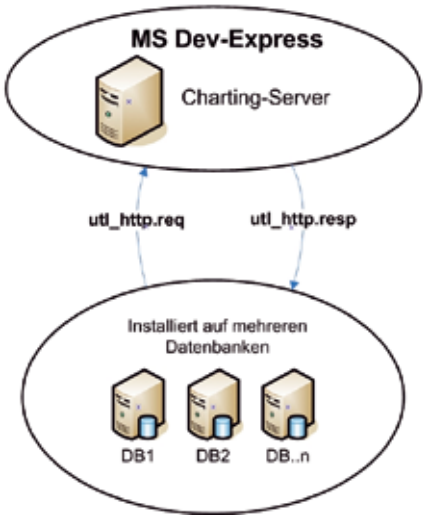

Abbildung 1: Kommunikation Datenbank – Charting-Server

	jFreeChart	Oracle Chart Builder	MS DevExpress	PL/PDF Chart
Technologie	Java	Oracle Java	ASP .NET	PL/PDF
Chart-Typen	+	–	++	– –
Ausgereiftheit	+	0	++	–
Layout	+		++	– –
Aktualität	++	Zuletzt 2002	++	. / .
Verbreitung	++	–		
Schnittstellen	Java	PL/SQL	.NET	PL/PDF
Support	+	–	+	
Kosten	Frei / 1124€ Doku (Global Site Lic.)	Frei	600 EUR pro Entwickler	ab 600 USD pro Datenbank

Tabelle 1: Die Entscheidung fiel zugunsten MS DevExpress

Nach Tests wird einer XML-Schnittstelle Vorrang eingeräumt. Eine Alternative wäre zum Beispiel eine Cursor-Lösung gewesen. Auf der .NET-Seite werden XML-Templates entworfen, die pro Chart-Typ die Steuerungsmöglichkeiten zum Aussehen des Charts bestimmen. So können Achsenbeschriftungen, Schriftgrößen, Skalierungen und einige Layout-Eigenschaften bestimmt werden (siehe Abbildung 2).

Auch die eigentlichen Chart-Daten werden über diese vordefinierte Schnittstelle transportiert, beim Performance-Chart beispielsweise über eine Serie von Punktepärchen (siehe Abbildung 3).

Dagegen sind einige Chart-Layout-Eigenschaften, wie die Hintergrund- oder Polylinien-Farben sowie auch die Schriftarten, vordefiniert und nicht übersteuerbar. Ein einheitliches Aussehen der erzeugten Charts, wo immer diese auch eingesetzt werden, wird dadurch garantiert (Corporate Design). Das gepostete XML wird deserialisiert und anhand des vorgegebenen Templates aus der XML-Struktur in.NET-Klassen umgesetzt. Diese .NET-Klassen sorgen für die entsprechende Befüllung des DevExpress-Charting-Objekts und geben dieses Objekt im PNG-Format zurück. Parallel wird auf der Datenbankseite ein PL/SQL-Package entwickelt, das mit .NET kommuniziert.

Die Hauptaufgabe besteht darin, die XML-Struktur zum angeforderten Chart zu generieren. Der Aufbau unterteilt sich in das Chart als Hauptobjekt, die Achsen, die beim Performance-Chart und dem Balkendiagramm notwendig sind, und die Serie, welche die eigentlichen Werte des Charts enthält. Bei einem Tortendiagramm werden zum Beispiel eindimensional die Anteile des Kuchens mitgegeben. Die Serien des Performance-Charts dagegen müssen aufeinanderfolgende Punktepärchen beinhalten. Das Performance-Chart muss mit zwei Serien befüllbar sein. Eine für den Werteverlauf des Portfolios und eine für einen Benchmark (siehe Abbildung 4).

Die Package-Prozeduren wandeln die übergebenen Attribute in XML-Tags um. Per HTTP-Request wird die aufgebaute XML-Struktur an die DevExpress-Komponente auf der .NET-Seite versendet. Am Ende steht der Empfang des Chart-Bilds als BLOB-Stream über den HTTP-Response. Dieses kann entweder in der Datenbank gespeichert, an eine aufrufende Web-Auskunftsapplikation übergeben oder gleich weiter ohne Speicherung in einen PL/PDF-Report eingebunden werden (siehe Abbildung 5).

```
  <Chart xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <Template>PMSPerformance2D</Template>
   <Height>1300</Height>
   <Width>1300</Width>
   <Title>Performance Chart</Title>
   <TitleFontSize>20</TitleFontSize>
   <LegendVisible>true</LegendVisible>
   <LegendFontSize>12</LegendFontSize>
   <GridInterlaced>false</GridInterlaced>
   <GridDisplay>3</GridDisplay>
   <GridAlignment>3</GridAlignment>
   <GridLineThickness>1</GridLineThickness>
   <Axes>
   ...
  </Axes>
```

Abbildung 2: XML-Template-Metadaten

```
<Series>
  <Serie>
   <Title>Portfolio 871600</Title>
   <ArgumentIsInAxisType>X</ArgumentIsInAxisType>
   <ValueIsInAxisType>Y</ValueIsInAxisType>
   <RGBColor>152</RGBColor>
   <DATAPoints>
     <DATA>
      <Argument>2010-01-01T00:00:00Z</Argument>
      <Value>100</Value>
 </DATA>
```

Abbildung 3: XML-Template-Punktedaten eines Performance-Chart

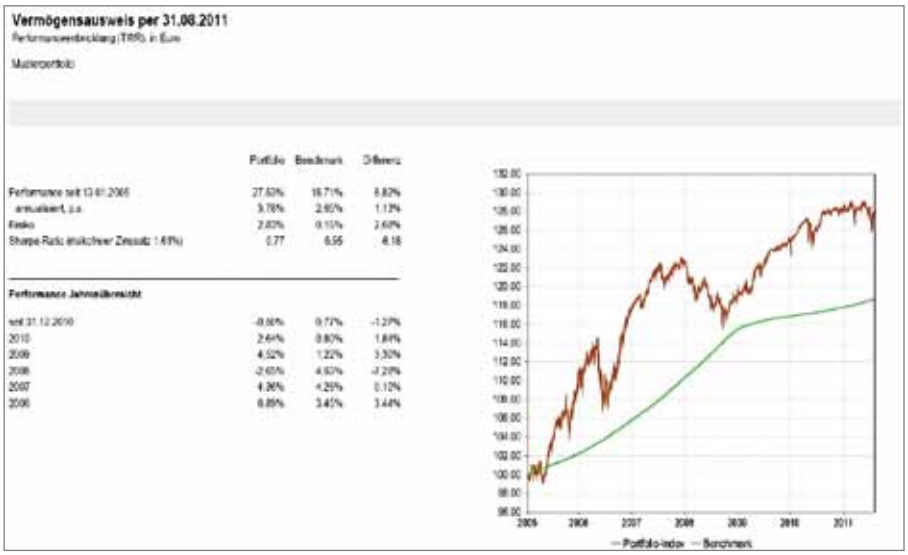

Abbildung 7: Prototyp

Fehler-Handling

Bei auftretenden Fehlern wurde das Augenmerk auf die Kommunikation zwischen Datenbank und Charting-Server gelegt. Wichtig war dabei unter anderem, wie das System reagiert, wenn der angesprochene Server nicht verfügbar ist. Für Oracle 10 hat sich leider herausgestellt, dass es in dem gegebenen Setup nicht möglich ist, ein „Connection Timeout“ abzufangen. Wenn der angesprochene Server nicht erreichbar ist, wartet die Datenbank endlos auf eine Rückantwort. Dieses Problem ist mit der Version 11 behoben; es wird nun eine entsprechende Timeout-Fehlermeldung durch UTL_HTTP zurückgeliefert.

```
   --Aufbau des Charts als Hauptobjekt
      SELECT XMLROOT(
            XMLELEMENT("Chart"
              , XMLATTRIBUTES('http://www.w3.org/2001/XMLSchema-in-
stance' AS "xmlns:xsi"
                              ,'http://www.w3.org/2001/XMLSchema' AS
"xmlns:xsd")
              , XMLELEMENT("Template", SELF.CHART_TYP)
              , CASE
                  WHEN SELF.DISPLAY_LANGUAGE IS NULL THEN NULL
                  ELSE XMLELEMENT("DispLang", SELF.DISPLAY_LANGUAGE)
                END
              , CASE
                  WHEN SELF.SHOW_COLORED_JN IS NULL THEN NULL
                  ELSE XMLELEMENT("ShowColored", global.comp_chart.
get_JN2Str(SELF.SHOW_COLORED_JN))
                END
              , CASE
                  WHEN SELF.INDEXED_JN IS NULL THEN NULL
                  ELSE XMLELEMENT("Indexed", global.comp_chart.get_
JN2Str(SELF.INDEXED_JN))
                END
              .
              .
              .
              .
              .
              , CASE
                  WHEN SELF.tab_axis IS NULL THEN NULL
                  ELSE XMLELEMENT("Axes",'')
                END
              , XMLELEMENT("Series",'')
            )
           , VERSION '1.0', STANDALONE YES
           )
      INTO v_xml
      FROM DUAL;

    --Achsen ergänzen
    IF SELF.tab_axis IS NOT NULL THEN
      vn_index:= SELF.tab_axis.FIRST;
      WHILE vn_index IS NOT NULL LOOP
        v_xml:= v_xml.appendChildXML('/Chart/Axes', SELF.tab_axis(vn_
index).Get_XML);
        vn_index:= SELF.tab_axis.NEXT(vn_index);
      END LOOP;
    END IF;

    -- Serien ergänzen
    vn_index:= SELF.tab_serie.FIRST;
    WHILE vn_index IS NOT NULL LOOP
      v_xml:= v_xml.appendChildXML('/Chart/Series', SELF.tab_
serie(vn_index));
      vn_index:= SELF.tab_serie.NEXT(vn_index);
    END LOOP;
```

Abbildung 4: Aufbau des Chart-Rahmens und Ergänzung der Achsen und Serien

Die Untersuchung der Anfragezeit bei verschiedenen Datenmengen war ein weiterer wichtiger Punkt in Sachen Stabilität. Die Theorie war, dass entweder eine Linearität zwischen der Punkteanzahl eines Performance-Charts und dessen Erzeugungszeit besteht oder dass die Zeit exponentiell steigt. Vielleicht gibt es auch Ausreißer oder Grenzwerte. Dazu ist ein Test-Package entwickelt worden, welches hintereinander Charts erzeugt, die immer mehr Punktepärchen enthalten, und dazu die Zeit misst (siehe Abbildung 6).

Die rote Kurve zeigt die Anfragezeit in Millisekunden, die verbraucht wurde, um die XML-Struktur zu versenden und das Bild zu empfangen. Die grüne Kurve dagegen bildet die Gesamtzeit ab, also inklusive der Vorbereitungszeit, die vergeht, bis ein SQL-Select die Punktedaten selektiert hat. Die Abteilung für Qualitätssicherung führte einen Lasttest unter realistischen und produktiven Bedingungen durch, der positiv ausfiel.

Prototyp

Der erfolgreichen Installation des Charting-Tools in der Oracle-Datenbank sollte die Migration eines bisher in Access erzeugten Reports mit einem Performance-Chart in einen PL/PDF-Report folgen. Die Implementierung des alphanumerischen Teils wurde bereits realisiert. Es fehlte nur noch das Anfordern und Einfügen des passenden Charts (siehe Abbildung 7).

Steffen Schumann
Berenberg Bank
steffen.schumann@berenbergbank.de

```
-- HTTP Request absetzen, Connection aufbauen
http_req := utl_http.BEGIN_REQUEST(pc_target_url, 'POST', utl_http.HTTP_VERSION_1_1);
utl_http.SET_HEADER(http_req, 'content-Type', 'text/xml');
utl_http.SET_HEADER(http_req, 'content-length', LENGTHB(self.chart_clob));

-- HTTP Response empfangen und verarbeiten
http_resp:= utl_http.GET_RESPONSE(http_req);

utl_http.get_header_by_name(http_resp, 'Content-Type', http_cont_type, 1);

IF http_cont_type LIKE 'Image%' THEN
BEGIN
  DBMS_LOB.CREATETEMPORARY(chart_blob, FALSE);
  --charting server hat ein Bild zurückgeliefert
  LOOP
    utl_http.READ_RAW(http_resp, response, 32767);
    DBMS_LOB.WRITEAPPEND(chart_blob, UTL_RAW.length(response), response);
  END LOOP;
EXCEPTION
  WHEN utl_http.END_OF_BODY THEN
    utl_http.END_RESPONSE(http_resp);
END;
ELSE
  --charting server hat KEIN Bild zurückgeliefert
  BEGIN
    --Die Rückmeldung vom Server wird an eine Variable übergeben,
    --die vom rufenden Programm ausgewertet werden kann.
    --Der Rückgabewert der Function GET_CHART_PICTURE bleibt NULL
    DBMS_LOB.CREATETEMPORARY(self.chart_fehler, FALSE);
    utl_http.READ_TEXT(http_resp, self.CHART_FEHLER, 30000);
    GLOBAL.PA_TRACE.TRACE( GLOBAL.PA_TRACE.TRACE_ERROR, methodenname, self.CHART_FEHLER);
  END;
END IF;
```

Abbildung 5: HTTP-Request und HTTP-Response

Abbildung 6: Zeitmessung (X-Achse: Zeit in MS, Y-Achse: Anzahl Punktepärchen in der XML-Struktur)