Verwaltung von multimedialen Objektdaten eines Veranstaltungsfundus mit Oracle und APEX

Petra Sauer
Beuth Hochschule für Technik Berlin

Schlüsselworte:
Datenbankentwurf, Datenmodellierungspattern, Oracle APEX, Bilddaten, XML-Daten

Einleitung

Für Theater, Museen oder Anbieter von kulturellen Veranstaltungen ist die Verwaltung von Gegenständen des Veranstaltungsfundus eine wichtige Aufgabe, die oftmals noch – vor allem bei kleineren Anbietern – mit Hilfe von Listen der Office-Pakete umgesetzt wird. Mit einem datenbankbasierten Verzeichnis von Fundusgegenständen für Veranstaltungen kann neben der Objektverwaltung auch das Reservierungs-, Ausleih-, Reparatur- und vor allem das Lagerbestandsmanagement unterstützt werden. Daneben ist eine Anbindung an weitere Datenbestände, wie beispielsweise der Auftrags-, Kunden- und Lieferantenverwaltung bis zum Facility Management möglich, so dass redundante Datenbestände reduziert und besser synchronisiert werden können.
Verzeichnisse der Fundusgegenstände werden in verschiedenen Ausprägungen benötigt, die oftmals ad hoc erstellt werden müssen. Verschiedene Klassifikationen müssen bedient und Selektionsmöglichkeiten angeboten werden, um eine anforderungsgenaue Suche zu realisieren. Eine weitere wichtige Anforderung ist die Verwaltung von Bildern zu den Objekten als Abbildungen des Gegenstands selbst wie auch zu dessen Identifizierung über einen QR-Code.
Im Beitrag wird die Entwicklung eines Katalogs von Fundusgegenständen für eine Freizeitanlage mit einem mehr als 10.000 Objekte umfassenden Veranstaltungsfundus vorgestellt. Die Entwicklung lief als studentische Projektarbeit eines Semesters und benutzte eine Oracle 11g Datenbank und APEX 4.0. Die Verwaltung der multimedialen Daten erfolgt über den Datentyp BLOB, wie auch zur Gegenüberstellung der Leistungsfähigkeit der objektrelationalen Erweiterung über den Datentyp ORD_IMAGE. Auf Basis von ORD_IMAGE wird die Arbeit mit QR-Codes gezeigt.

Anforderungsanalyse

Das FEZ Berlin ist – bezogen auf die Fläche - einer der größten Freizeitparks in Europa und bietet ganzjährlich Veranstaltungen für Kinder und Jugendliche an. Veranstaltungen werden im Gebäude sowie im Park selbst durchgeführt. Veranstaltungen finden einmalig, aber auch in einem festen zeitlichen Rhythmus wiederkehrend statt und sind mit thematisch-orientierten oder auch universell einsetzbaren Gegenständen des Veranstaltungsfundus ausgestattet. Dieser ist über die Jahre gewachsen und wird ständig erweitert und erneuert, aber auch an andere Einrichtungen teilweise verliehen. Während noch vor einigen Jahren handschriftliche Listen, Word-Dokumente oder Excel-Dateien für die Verwaltung der Fundusgegenstände eingesetzt wurden, führten jüngere Anforderungen nach Hinterlegung von Bildern zum Einsatz einer Bildergalerie. Jedem Fundusobjekt ist dort ein Bild zugeordnet. Neben dem Verweis auf die Bilddatei sind Sachdaten zum Fundusobjekt wie Name, Maße, Anzahl, Lagerort als textuelle Daten hinterlegt. Obwohl die Handhabung der Bildergalerie intuitiv und nutzerfreundlich ist, existieren gravierende Mängel in der verfügbaren Funktionalität, die teilweise die Benutzung von Dummy-Objekten oder auch redundante Datenablage erfordern. Folgende

Minimal-Anforderungen existieren seitens der Nutzer an eine effiziente Verwaltung der Fundusobjekte:

- Die Verwaltung von **Bild- und Sachdaten** soll unabhängig erfolgen. Eine dynamische Anzahl von Bildern soll zu einem Fundusobjekt hinterlegt werden können.
- Eine Verwaltung und Zuordnung von **Lagerorten** soll möglich sein.
- Die **Ausleihe und Reservierung** sowie die Verwaltung der Kunden sollen unterstützt werden.
- Die **Reparatur von Fundusobjekten** und die Erstellung von **Reparaturaufträgen** sollen umgesetzt werden. Die **Reparaturfirmen** sollen verwaltet werden können und eine Unterscheidung in externe Firmen und eigene Handwerker soll erfolgen.
- Die Zuordnung zu einer beliebigen Anzahl von **Veranstaltungen** soll zeitraumbezogen verwaltet werden.
- Ein aus einer Liste auswählbarer **Status** ist einem Fundusobjekt zuordenbar. Defaultwerte sollen genutzt werden.
- Eine Kategorisierung und **Verschlagwortung** des Fundusbestands soll realisiert werden.
- Sowohl über Schlagwörter als auch über die weiteren Sachdaten zum Fundusobjekt soll eine Suche erfolgen.
- Eine Kennzeichnung in Außen- und Innenmobiliar soll erfolgen.

Weitere Anforderungen für eine zukünftige Weiterentwicklung eines Funduskatalogs sind:

- Der Standort eines Fundusobjektes soll verfolgt werden und evtl. auf einer Karte beispielsweise des OpenStreetMap-Projektes dargestellt werden können. Dazu sollen eine Geocodierung auf Basis der Adressdaten der Kunden, Reparaturfirmen und Veranstaltungsorte erfolgen und in Abhängigkeit vom Objektstatus ausgewertet werden.
- Änderungen des Objektstatus sollen verfolgt und aus einem Zustandsgraph abgeleitet werden können. Als ein neuer Status soll die Sperre eines zurückgegebenen Fundusobjektes für eine Ausleihe infolge von Reparatur- oder Reinigungsaufgaben einbezogen werden.
- Aspekte wie die Materialzusammensetzung sollen künftig in Zusammenhang mit einer Brandlastberechnung sinnvoll genutzt werden. Eine Ergänzung der Daten der Materialzusammensetzung von Fundusobjekten durch einen Brandlastkoeffizienten sollte entwickelt werden.
- Wünschenswert ist langfristig die Auszeichnung der Fundusobjekte mit einem **Barcode** zur Identifizierung und evtl. auch Hinterlegung weiterer wichtiger Daten wie Lagerort, Anschaffungsdatum, Wartungsintervall etc.

Aus der Analyse der Anforderungen resultierten die folgenden, wesentlichen Bereiche, die beim Datenbankentwurf abgedeckt werden mussten:

- **Objektdaten** liefern Informationen zum Fundusgegenstand und dessen Anschaffung, wie z.B. Maße, Name, Anschaffungsdatum, Verantwortlichkeit. Als Objektdaten werden auch die **Bilddaten** des Objektes betrachtet. Sie sind direkt an das Objekt gebunden und von seiner Existenz abhängig.
- **Materialdaten** umfassen verwendete Materialien, deren Zusammensetzung und Beschreibung und können flexibel erweitert werden bis hin zur Zuordnung von Brandlastkoeffizienten.
- **Lagerdaten** beinhalten Daten zu Lagerort, Raum, Gebäude, Adresse und können erweitert werden um geografische Koordinaten.
- Als **Klassifikationsdaten** wurden Daten zur Kategorisierung und Verschlagwortung hinterlegt.
- **Reparaturdaten** erstrecken sich auf die eigentliche Reparatur sowie die ausführende Institution, die intern oder extern sein kann.
- Als **Veranstaltungsdaten** sind Name, Ort und Zeitdauer der Veranstaltung zu verwalten.
- **Reservierungs- und Verleihdaten** sollen ebenfalls die Kundendaten umfassen.

Die Implementierung erfolgte mit Oracle 11g Enterprise Edition. Die Nutzung ist auf einer Oracle 11g Express Edition (XE) geplant. Neben den alphanumerischen Daten, die über die Basisdatentypen abgebildet werden, mussten Entscheidungen zur Verwaltung der Bilddaten getroffen werden.

Für die Verwaltung der Bilddaten zu Fundusobjekten in der Datenbank wurden die folgenden Datentypen gegenübergestellt:
- der Datentyp BLOB, der Binärdaten bis zu einer Größe von 4 GB speichern kann, und
- der Objekttyp ORD_IMAGE.

Da der Objekttyp die Binärdaten in einem Attribut vom Datentyp BLOB speichert, aber zusätzlich die Metadaten zum Bild vorhalten kann, ist der Objekttyp ORD_IMAGE ideal für den Einsatz zur Erzeugung von Barcodes als QR-Code. Da der Einsatz auf einer Oracle-XE-Instanz erfolgen soll, wurde trotz der Vorteile von ORD_IMAGE auf BLOB zurückgegriffen.

Datenbankentwurf

Der Datenbankentwurf sollte die Stammdatenverwaltung über wiederverwendbare Datenstrukturen und Pattern realisieren. Als Entwurfsaspekte wurden u.a. beachtet die Vermeidung von Redundanzen, die Realisierung der erforderlichen Detailliertheit der Objektbeschreibung, Beachtung von Versionierungswünschen, Umsetzung von Klassifikationen und Kategorisierungen.
Aus verschiedenen, nicht disjunkten Teilentwürfen wurde ein Integrationsmodell abgeleitet und implementiert. Dabei wurde der Betrachtung der Qualität und Integrität des Datenmodells große Beachtung geschenkt. Die typischen Fehler im Bereich der Datenmodellierung wurden herausgearbeitet und analysiert.

Typische Modellierungsfehler, die sowohl aus der Analyse dieser als auch weiterer Projektarbeiten identifiziert werden konnten, betrafen die folgenden Bereiche:
- Typ-Instanz-Erkennung,
- Teil-Komponenten-Erkennung,
- Erkennung der Quantifizierung eines Beziehungstyps,
- Erkennung der Qualifizierung eines Beziehungstyps,
- Erkennung der Qualifizierung eines Objekttyps.

Am Beispiel der Fehler bei der Typ-Instanz-Erkennung werden nachfolgend einige typische Ausprägungen herausgearbeitet und systematisiert. Die weiteren, genannten Fehlerarten werden in diesem Beitrag nicht detailliert beschrieben.
Als Use Case wird die Betrachtung des Fundusobjektes herangezogen: Im Anwendungsbeispiel existieren typischerweise mehrere Exemplare eines Fundusobjektes (Datenmodell in Abb. 1), beispielsweise 32 Stellwände mit Stoffbespannung. Sie haben die gleichen Maße, den gleichen Lagerort, die gleiche Beschreibung. Die konkreten Stellwände können in Reparatur, in Benutzung in einer Veranstaltung oder am Lager sein.

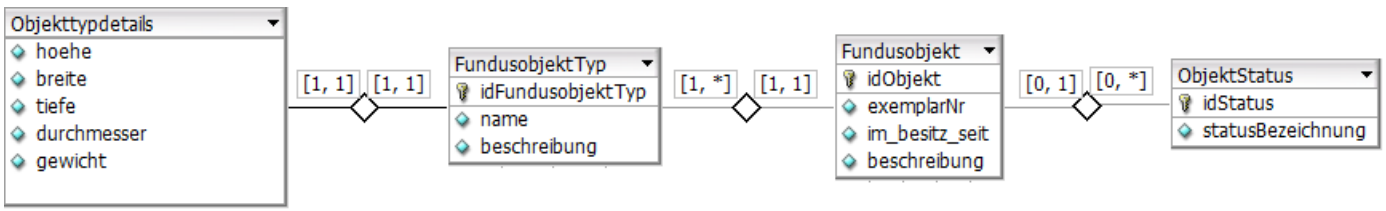

Abbildung 1: Datenmodellausschnitt Typ-Instanz von Fundusgegenständen

Als identifizierte Modellierungsfehler traten auf:

- Der Typ wird nicht erkannt. Es wird nur mit dem Einzelobjekt gearbeitet (Datenmodell in Abb. 2).
  Resultat: Alle Informationen und Beziehungen werden dem einzelnen Objekt zugeordnet.
  Problem: Es treten massive Redundanz- und damit Konsistenzprobleme auf. Diese betreffen die typbezogenen Eigenschaften und Beziehungen. Im Anwendungsfall werden jedem Objekt die Maße sowie die Liste der Schlagwörter und der Materialien zugeordnet.

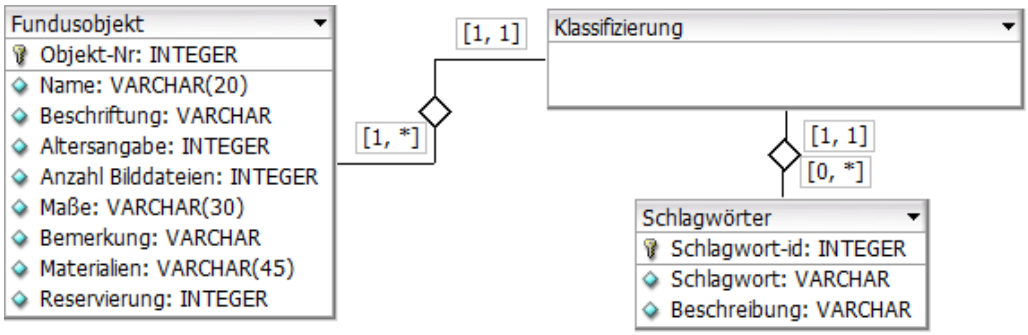

Abbildung 2: Datenmodell ohne Fundusobjekttyp

- Typ und Instanz werden Eigenschaften nicht korrekt zugeordnet. Die möglichen Fehler betreffen:
  - Eine zu starke Detaillierung.
    Resultat: Eigenschaften und Beziehungen des Typs werden dem Exemplar zugeordnet.
    Problem: Es treten Redundanzprobleme auf. Daten werden unnötig erhoben und gespeichert. Beispielsweise werden die Informationen zum Lagerort dem einzelnen Fundusobjekt zugeordnet und für alle Objekte desgleichen Typs redundant abgelegt.
  - Eine zu starke Generalisierung.
    Resultat: Eigenschaften des Exemplars werden dem Typ zugeordnet.
    Problem: Informationen gehen verloren. Wird beispielsweise dem Fundusobjekttyp die Beziehung zum Verleih zugeordnet, kann nicht hinterlegt werden, welches konkrete Fundusobjekt wohin ausgeliehen ist.
  - Der Typ wird nur zur Klassifikation benutzt, typbezogene Eigenschaften und Beziehungen werden dem Exemplar zugeordnet. Die Resultate und Probleme entsprechen dem Phänomen der zu starken Detaillierung (Modell in Abb. 3).

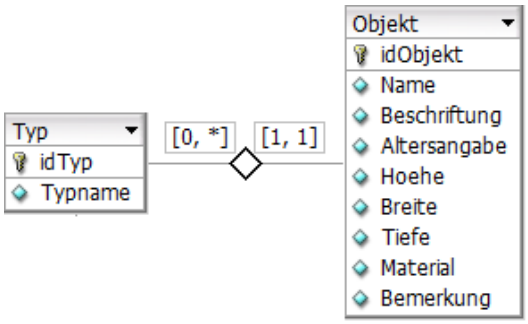

Abbildung 3: Datenmodell mit Typ zur Klassifizierung der Fundusobjekte

- Der Typ wird erkannt, aber nur implizit umgesetzt. Im Anwendungsfall wird nur der Fundusgegenstand modelliert. Ihm wird als Attribut die Anzahl der Exemplare zugeordnet.
  Resultat: Die Eigenschaften und Beziehungen des Exemplars werden dem Typ zugeordnet.
  Problem Es treten Probleme der Interpretation von Daten auf. Beispielsweise kann der Status nur als Status der gesamten Objektmenge interpretiert werden, obwohl je Exemplar hinterlegt sein müsste, welchen konkreten Status dieses aktuell aufweist (Datenmodell in Abb. 4).
  Problem: Es treten Probleme mit abhängigen Werten auf. Beispielsweise wird die Gesamtanzahl beim Objekt hinterlegt und die Anzahl der ausgeliehenen Exemplare bei Ausleihe, die Anzahl der aktuell in Reparatur befindlichen Exemplare bei Reparatur etc. Diese abhängigen Werte müssen entsprechend synchronisiert werden, was in der Implementierung über Trigger erfolgen kann.

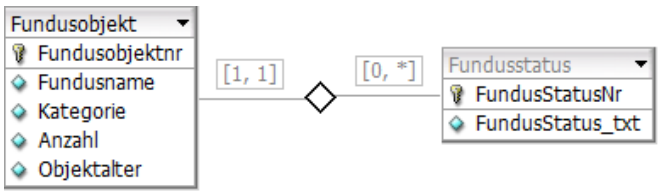

Abbildung 4: Datenmodell mit implizitem Fundusobjekttyp

Datenmodellierungspattern

Um derartige Modellierungsfehler künftig zu reduzieren wird auf die Verwendung von allgemeingültigen, wiederverwendbaren Datenstrukturen in Form von Datenmodellierungspattern orientiert. Dazu wurden die Datenstrukturen des Projektes sowie weiterer Projekte verglichen und u.a. die folgenden Pattern identifiziert. Dabei erfolgt eine Unterscheidung nach Kontextbezug vs. Allgemeingültigkeit.

Als kontextunabhängige, allgemeingültige Strukturpattern wurden definiert:
- **Kompositionspattern** – Um Redundanz im Modell und im Datenbestand zu vermeiden wird die Zusammensetzung von Fundusobjekten aus weiteren Fundusobjekten über eine rekursive Beziehung abgebildet.
- **Partitionierungspattern vertikal** – Die Eigenschaften eines Objekttyps wurden nach der angenommenen Häufigkeit des Zugriffs auf zwei Objekttypen aufgeteilt, die über eine 1:1-Beziehung verknüpft sind.
- **Typ-Instanz-Pattern** – Existieren mehrere Instanzen eines Objektes, so muss nach Eigenschaften des Typs und Eigenschaften der Instanz unterschieden und den gebildeten Objekttypen des Typs und der Instanz zugeordnet werden. Analog müssen die Beziehungen zu weiteren Objekttypen nach Beziehungen des Typs und der Instanz unterschieden werden.

Als kontextbezogene Datenmodellierungspattern wurden u.a. gefunden und definiert:
- **Adresspattern** – Die Adressangaben werden als eigener Objekttyp modelliert und zu demjenigen Objekttyp in Beziehung gesetzt, dessen Lokationsdaten verwaltet werden sollen. Als optionaler Bestandteil kann die geografische Koordinate über den Objekttyp SDO_GEOMETRY abgebildet werden.
- **Auftragspattern** – Die Auftragsdaten werden im Sinne der Redundanzreduzierung und Normalisierung in Auftragskopf- und Auftragspositionsdaten aufgeteilt. Der Auftragnehmer wird über einen Beziehungstyp dem Auftragskopf zugeordnet. Für dessen Adressdatenverwaltung wird das Adresspattern eingesetzt.

- **Bilddatenpattern** – Die Bilddaten eines Objektes werden in einem eigenen Objekttyp modelliert. Bildname, Auflösung, Datum, Maße und die Bilddaten selbst oder die Information über den Verzeichnis- und Dateinamen bei externer Speicherung gehören dazu. Für dessen Umsetzung ist daneben zu entscheiden, ob eine relationale oder objektrelationale Implementierung erfolgen soll, d.h. sollen die Meta- und die Bilddaten in einem gemeinsamen Objekttyp verwaltet werden oder sollen die Metadaten neben den Bilddaten verwaltet und durch den Nutzer konsistent gehalten werden.

Verwaltung der Bilddaten für QR-Codes in Oracle

Auf Basis des Objekttyps ORD_IMAGE wurde prototypisch die Arbeit mit QR-Codes umgesetzt. Die Erzeugung der QR-Codes zur Objektidentifizierung erfolgt über einen Webserviceaufruf, zu dem das PL/SQL-Paket UTL_HTTP eingesetzt wurde. Aus einer übergebenen Zeichenkette wird über einen Trigger der QR-Code erzeugt und neben den Sachdaten des Fundusobjektes als ORD_IMAGE-Objekt in der DB abgelegt ([Heus10]).

Applikationsentwicklung mit APEX 4.0

Die Applikationsentwicklung wurde mit APEX 4.0 auf einem eigenen APEX-Server umgesetzt, auf dem jedes studentische Team einen eigenen Workspace erhielt. Allen Workspaces war ein gemeinsames Datenbankschema zugeordnet, in dem Testdaten zur Fundusverwaltung vorgehalten wurden und welches aus der Implementierung des integrierten Datenmodells resultierte. Neben der Stammdatenverwaltung wurden Visualisierungen des Fundusbestands nach verschiedenen Kriterien vorgenommen. Zusätzlich wurde mit verschiedenen Diagrammtechniken in Apex gearbeitet, um die Verteilung beispielsweise nach Veranstaltungszugehörigkeit, nach dem Objektstatus oder auch den Lagerorten sichtbar zu machen. Es sollten lesende und schreibende Zugriffe unterstützt werden, so dass folgende Anforderungen gestellt wurden:

- Ausgehend von definierten Ausschnitten des ER-Diagramms sollten Formulare bereitgestellt werden. Dabei sollten die Möglichkeiten der Arbeit mit Wertelisten, Validierungen, Formatierungen und Defaultwerten umfassend genutzt werden ([Czar11]).
- Um eine größtmögliche Vielfalt der Benutzerinteraktion zu erreichen, sollten zur Unterstützung der lesenden Zugriffe interaktive Berichte für Endnutzer erstellt und für ausgewählte Endnutzer angepasste, interaktive Berichte beispielhaft vorgefertigt werden.
- Für ausgewählte Daten sollten Diagramme erstellt werden.

Neben der Umsetzung der Basisvorgaben wurde eine Vielzahl weiterer Features zielgerichtet eingesetzt.

Der **Kalender** wurde sehr häufig genutzt, da er out-of-the-box in Regionen einsetzbar ist. Interessante Anwendungen ergaben sich, indem der Kalender als zentrales Element einer Portalseite positioniert (Abb. 5) und damit ein zentraler Überblick über Veranstaltungen realisiert wurde.

Zur Information der verantwortlichen Mitarbeiter über das Auftreten definierter Ereignisse, wie beispielsweise die Ausleihe oder Rückgabe eines Fundusobjektes, wurde ein **Mailversand** eingerichtet. Die Umsetzung erfolgte in Java.

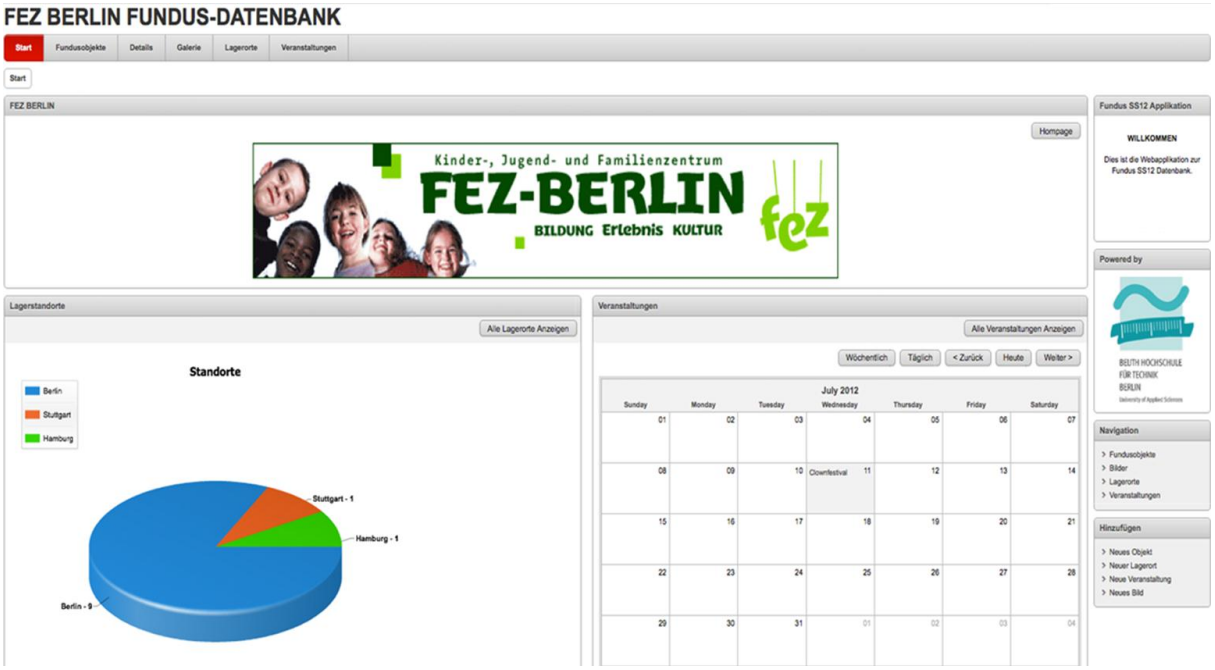

Abbildung 5: Nutzung von Kalender und Diagramm auf einer Portalseite

Als interessante Anwendung für ein **Diagramm** wurde die Redundanzanalyse von Zeichenketten einer Tabellenspalte entwickelt. Da teilweise auf die Vorgabe von Stammdaten verzichtet wurde und Schreibzugriffe möglich waren, wurden beispielsweise redundante und fehlerhaft bezeichnete Statusangaben erfasst. Das entwickelte Diagramm visualisiert die Häufigkeit des Eintrags von Statusangaben.

Die Arbeit mit **Bildern**, die lediglich als BLOB-Datentyp verfügbar waren und Erstellung von Alben wurde in verschiedensten Varianten umgesetzt. Insbesondere die Skalierung der Bilder wurde sehr verschieden implementiert, u.a. mit HTML-Einbettung (Abb. 6).

Region Source

```
select distinct
b.name,
"IDBILD",
"IDFUNDUSOBJEKTTYP_FK",
"DATEINAME",
"AUFLOESUNG",
"ALBUMNAME",
'<img style="border: 2px solid #D40000; -moz-border-radius: 2px; -webkit-border-
radius: 2px;" '||
    'src="'||apex_util.get_blob_file_src('P5_BILD', "IDBILD") ||
'"height="150" width="150" />' "BILD"
from "BILD", fundusobjekttyp b
where "IDFUNDUSOBJEKTTYP_FK" = b.idfundusobjekttyp
and "BILD" is not null;
```

Abbildung 6: Select-Statement mit eingebettetem HTML-Skript zur Skalierung von Bildern

Weitere interessante Anwendungen betrafen **Formulare** und **Berichte** und fokussierten sich u.a. auf das Reservierungsmanagement, das Lagerbestandsmanagement oder auch eine Raum-Gebäude-Objekt-Verwaltung - als rudimentäre Abbildung eines Facility Management Systems.

Fazit

Die Entwicklung eines Katalogs von Fundusobjekten ist eine mehr als adäquate Alternative zu den meist durch die Nutzung von Office-Paketen geprägten Listen oder Tabellen und kann in relativ kurzer Zeit mit einem Rapid-Application-Framework wie APEX umgesetzt werden. Voraussetzung ist ein durchdachter Datenbankentwurf, der durch die Arbeit mit Datenmodellierungspattern Fehler reduzieren und eine gute Qualität erreichen kann. Für die zielgerichtete Nutzung von APEX sind der Entwurf der zu implementierenden Seiten und deren Navigation unerlässlich. Der Seitenentwurf ist ebenfalls ein wesentliches Element für eine Arbeitsteilung im Implementierungsteam.
Trotz der großen Entwicklungsfortschritte, sind bei der Arbeit mit APEX einige Probleme aufgetreten:

- Viele Projektteams kritisierten den hohen Einarbeitungsaufwand und die sehr langsame Lernkurve.
- Das Benutzerinterface wurde als nicht intuitiv zugänglich eingeschätzt. Die Möglichkeit, über verschiedene Wege durch das Interface, Aufgaben gleichwertig umzusetzen, wurde teilweise als störend empfunden.
- Während einfache Seiten sehr schnell umgesetzt werden können, sind aufwändige Seiten oder Regionen sehr kompliziert zu entwickeln.

Insgesamt wurden in sehr kurzer Zeit tolle Projekte umgesetzt, die die potenziellen Anwender stark beeindruckten und viele neue Ideen zur Weiterentwicklung formulieren ließen.

Quellen

[Czar11] Czarski, Carsten: Oracle Application Express. Database Pro 02/2011, S. 90 – 94;

[Heus10] Heuschkel, Steffen; Sauer, Petra; Herrmann, Frank: Management von Geo- und Standortdaten in Freizeitanlagen mit Oracle-Technologien. DOAG-Konferenz, Nürnberg, 2010;

Kontaktadresse:

Prof. Dr. Petra Sauer
Beuth Hochschule für Technik Berlin
Luxemburger Straße 10
D-13533 Berlin

Telefon: +49(0)30-4504 2691
E-Mail: sauer@beuth-hochschule.de
Internet: prof.beuth-hochschule.de/~sauer