Tabular Forms de luxe mit APEX Collections

Andreas Wismann
WHEN OTHERS
Kaarst, NRW

Schlüsselworte:

APEX Collections, PL/SQL, Tabellarische Formulare, MRU

Einleitung

Tabellarische Formulare in Oracle Application Express 4.1 basieren auf einer robusten, assistentengesteuerten Engine, mit denen datensatzbasierende Aufgaben effizient umgesetzt werden können. Oft benötigt man in anspruchsvolleren Projekten jedoch größere Kontrolle über die eingegebenen Daten und deren Verarbeitung - und sei es nur, um mehr als eine einzige MRU-Region pro APEX-Seite abbilden zu können. Ab diesem Punkt führt kein Weg mehr an individueller PL/SQL-Programmierung vorbei. Die manuelle Behandlung der Ein- und Ausgabe stellt sich häufig als Produktivitätsbremse heraus, insbesondere wenn jeder Prozess erneut auf der berühmten "Grünen Wiese" beginnt. Für den Benutzer ergibt sich daraus selten eine wahrnehmbare Verbesserung der Anwendung.

Hier kommen nun die APEX Collections ins Spiel. Deren mächtige API ermöglicht Features wie Persistenz (tabellarische Benutzereingaben können von einer APEX-Seite zur nächsten weitergegeben werden), Eingabepufferung (Daten lassen sich beliebig oft zwischenpeichern, bevor sie gegen fachliche Tabellen festgeschrieben werden) und feingranulare Transaktionssteuerung (ein Teil der Datensätze kann committen, während der Rest zur Bearbeitung an den Benutzer zurückgeht). Sie basieren auf quasi „privaten“ Datenbanktabellen, deren Inhalte nur für die jeweilige APEX-Session sichtbar sind; der Entwickler braucht sich also nicht mit dem Overhead der Sitzungsverwaltung auseinanderzusetzen.

In diesem Manuskript konzentriere ich mich auf den technologischen Hintergrund der APEX Collections. Mein Vortrag geht schwerpunktmäßig auf die Nutzungsaspekte ein und zeigt, wie Sie Collections in Ihren Anwendungen einsetzen können. Dabei werden verblüffende neue Features in APEX-Anwendungen demonstriert, die der Benutzer tatsächlich positiv wahrnimmt. Sie erfahren, wie Sie solche Verbesserungen mit einem überschaubaren Aufwand an Individualprogrammierung erreichen können.

Tabellarische Formulare („Tabular Forms“)

Mit Hilfe einer APEX-Anwendung lassen sich Datenbank-Tabellen bekanntermaßen einfach und komfortabel editieren. Es funktioniert wie seinerzeit in Oracle Forms: Als Entwickler deklariere man einen Block (in APEX: eine Region) und wähle die relevanten Tabellenspalten aus, die der Anwendungsbenutzer lesen und bearbeiten darf. Der Speichern-Prozess feuert letztendlich das generierte DML gegen die Tabellen und committet die Transaktion.

Dieses Szenario ist sehr effizient zu modellieren und macht den Reiz von Oracle Application Express aus: Rapid Application Development von seiner Schokoladenseite, fast schon wie „Programmierung ohne Programmierung“. Mit keinem anderen Tool aus dem Oracle-Werkzeugkasten löst man solche

Aufgaben schneller als mit APEX. Kunden sind zu Recht beeindruckt, wenn schon wenige Stunden nach der Anforderungsanalyse eine vollständig bedienbare APEX-Applikation präsentiert wird, die nebenbei auch noch sehr ansehnlich daherkommt. Oft stellt sich der Prototyp bereits als so gut heraus, dass er ohne nennenswerte Modifikation in den Produktionsbetrieb übernommen wird.

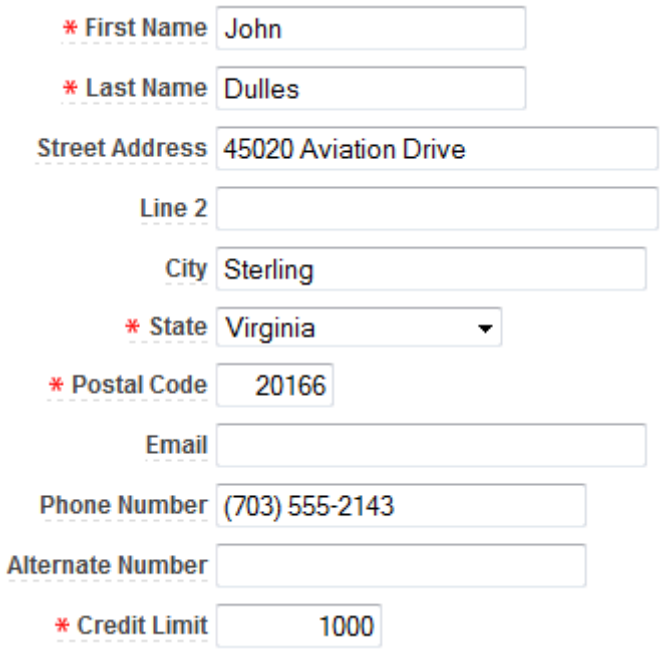
Aber man muss auch zugeben: Dieser Typ Anwendung ist nicht besonders komplex. Dass APEX dermaßen „schnell in Fahrt“ kommt und dabei so stark punktet führt manchmal dazu, dass es nicht als Entwicklungsplattform für weit anspruchsvollere Dinge wahrgenommen wird.

Wie ändern sich also die Bedingungen, wenn die Herausforderungen zunehmen? Etwa diese:

- Das Datenmodell wird erst in einem sehr späten Projektabschnitt verabschiedet, während man die Formulare (aus fachlicher Sicht) schon viel früher bauen könnte
- Die APEX-Anwendung darf nicht direkt in die Produktionstabellen schreiben
- Die Transaktion wird nicht durch Ausfüllen einer einzelnen Seite abgeschlossen, sondern umfasst mehrere Einzelseiten, die ihre Ergebnisse zwischenspeichern müssen (wie in einem Warenkorb-System)
- Der Anwender soll einzelne Bedienschritte rückgängig machen können, selbst nachdem er schon gespeichert hat - und zwar ohne optionale Datenbank-Features wie Flashback
- Es sind Views im Spiel, die nicht einfach per DML beschrieben werden können
- Die Daten sind von komplexer Natur, etwa Nested Tables oder XMLTYPE

Einzeldatensatz-Verarbeitung

Bei der Verwendung von **Single-Row-Items**, wo jedes Eingabefeld auf der APEX-Seite einem Attribut eines Datensatzes in einer Tabelle entspricht, kann der Entwickler bequem mit dem so genannten Session Cache arbeiten. Das Prinzip: Pro Item auf jeder Anwendungsseite existiert genau eine globale Variable, die von APEX automatisch beim Anlegen des Items erzeugt und passend benannt wird. Der Session Cache (oder: Session State) „trägt“ die Feldinformation in dieser Variablen dann so lange weiter, wie der Benutzer im selben Browserfenster eingeloggt bleibt (streng genommen sogar noch darüber hinaus), selbst wenn er die Anwendungsseite wechselt.



Wann und wohin die Benutzereingaben weggeschrieben werden, lässt sich über Prozesse und Bindevariablen von überall in der Anwendung flexibel festlegen („`update tabelle set spalte = :P4711_benutzereingabe where key = :P4711_hidden_key`“). Hierbei bietet die APEX-Engine dem Entwickler maximalen Komfort.

Verarbeitung mehrerer Datensätze

Ein völlig anderes Bild ergibt sich, sobald in **Multi-Row-Items** in tabellarischen Formularen („Tabular Forms“) zum Einsatz kommen. Also typischerweise dann, wenn auf der Benutzeroberfläche mehrere Datensätze in einem Tabellenraster präsentiert werden.

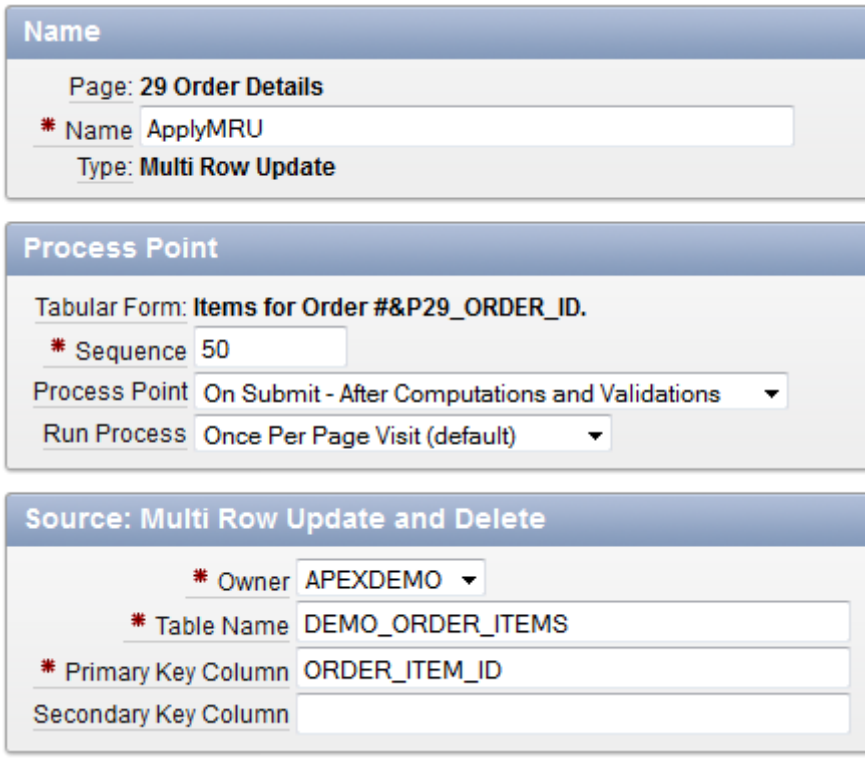
Product Name	Quantity	Unit Price	Extended Price	Product Image
Blouse [$60]	4	$60.00	$240.00	
Skirt [$80]	3	$80.00	$240.00	
Bag [$125]	2	$125.00	$250.00	
			$730.00	

Das klingt zwar wie der Standardfall einer Datenbankanwendung, APEX-intern wird das jedoch weniger elegant gehandhabt als man vielleicht vermuten würde. Das Assistenten-gesteuerte Erstellen eines tabellarischen Formulars verläuft für den Entwickler zunächst scheinbar mühelos: Erstellen einer SQL Report Region, Formulieren des SQL-Statements, Feinschliff der Report-Darstellung.

Region Source

```
select oi.order_item_id,
       oi.order_id,
       oi.product_id,
       oi.unit_price,
       oi.quantity,
       (oi.unit_price * oi.quantity) extended_price,
       dbms_lob.getlength(product_image) product_image ,
decode(nvl(dbms_lob.getlength(pi.product_image),0),0,null,
    '<img style="border: 4px solid #CCC; -moz-border-radius: 4px; -webkit-border-
radius: 4px;" '||
    'src="'||
    apex_util.get_blob_file_src('P6_PRODUCT_IMAGE',pi.product_id)||
    '" height="75" width="75" alt="Product Image" title="Product Image" />')
    detail_img
from DEMO_ORDER_ITEMS oi, DEMO_PRODUCT_INFO pi
where oi.ORDER_ID = :P29_ORDER_ID
and oi.product_id = pi.product_id (+)
```

Die nötigen DML-Prozesse zum Speichern legt APEX dann komplett automatisch an (hier spricht man von „MRU“-Prozessen, was für Multi Row Update steht).



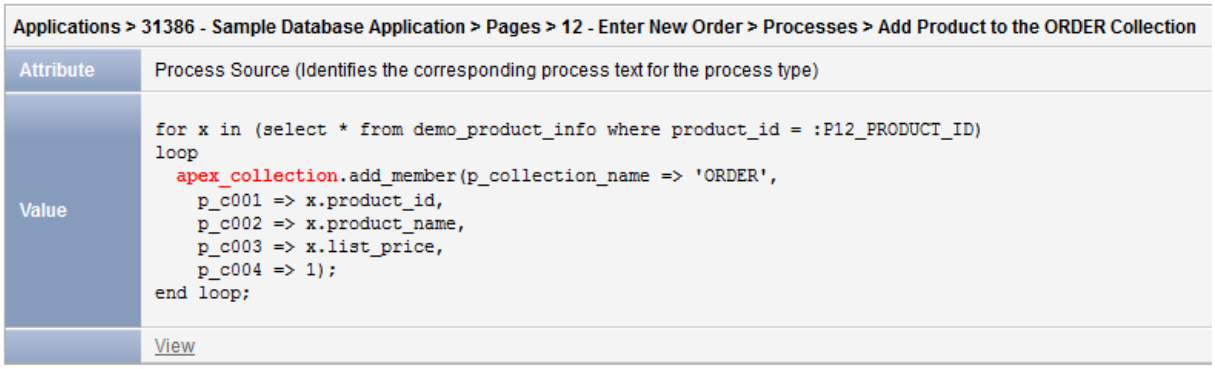
Der Pferdefuß liegt hier: Weicht der Programmierer auch nur ein Jota von diesem automatischen Verfahren ab, beginnt für ihn die Handarbeit.

1. Der Assistent kann pro APEX-Seite immer nur ein einziges tabellarisches Formular anlegen. Benötigt man zwei oder mehr Tabellen, muss jede (auch die erste) per manueller PL/SQL-Programmierung erstellt werden.
2. Die automatisch erzeugten MRU-Prozesse tun ihre Pflicht, nicht mehr und nicht weniger. Sollen sie Geschäftslogik implementieren, muss man sie ersetzen und neu programmieren.
3. Wechselt der Benutzer die Seite im Browser, kann der Entwickler anschließend programmatisch nicht mehr auf die tabellarischen Formulare der vorigen Seite zugreifen. Die Eingabefelder in tabellarischen Formularen besitzen nämlich im Gegensatz zu Single Row Items keinen Session State.

APEX Collections

Das APEX Application Programming Interface (API) stellt dem Programmierer das PL/SQL-Package `APEX_COLLECTION` zur Verfügung. Dazu gehören die Tabellen `WWV_FLOW_COLLECTIONS$` und `WWV_FLOW_COLLECTION_MEMBERS$` sowie eine View namens `APEX_COLLECTIONS`. Auf Letztere greift man beim Programmieren am häufigsten zu. Zusammen bezeichnet man das Konzept als APEX Collections. Es handelt sich um ein mächtiges Instrumentarium zur programmatischen Verarbeitung tabellarischer Daten in APEX, das nicht mit PL/SQL Collections (als Bestandteil der Programmiersprache) verwechselt werden sollte.

Einen Eindruck einer Anwendung, die sowohl mit „klassischen“ MRUs als auch mit APEX Collections arbeitet, können Sie sich in der Demo-Applikation verschaffen, die zusammen mit der APEX-Installation (bzw. bei der Neuanlage eines APEX-Accounts unter apex.oracle.com) angelegt wird. Dort wird ein einfaches Shop-System vorgestellt, aus dem auch die oben gezeigten Abbildungen stammen.



Codebeispiel aus der „Sample Database Application“

Schritt1: Collection initialisieren

```
apex_collection.create_collection_from_query(
  p_collection_name => 'meine_collection',
  p_query           => 'select spalte_A, spalte_B ' ||
                       '  from tabelle_oder_view ' ||
                       ' where bedingungen'
)
```

Man übergibt dem create-Befehl also ein SQL-Statement, das zur Laufzeit ein Resultset zurückgibt. Der Clou ist, die WHERE-Bedingungen dabei so präzise zu formulieren, dass genau diejenigen Datensätze selektiert werden, die der Anwender im jeweiligen Arbeitsschritt benötigt. Denn der Befehl filtert und „materialisiert“ genau diese Datensätze unter dem angegebenen Namen in die Collection-Tabelle. Die Collection sollte also stets „auf Maß“ erzeugt werden – es wäre wenig sinnvoll, beispielsweise alle jemals vom Kunden bestellten Artikel in die Collection zu schreiben und sie erst in der Anwendungsseite (erneut!) zu filtern. Oder – noch schlimmer, aber bereits vorgekommen – alle jemals bestellten Artikel *aller* Kunden… Kurz gesagt: Wenn Sie merken, dass beim Anlegen der Collection mehrere Sekunden ins Land gehen, dann ist entweder Ihr SQL extrem unperformant oder Sie selektieren bei weitem zu viele Datensätze (oder gar beides ;-)

Schritt 2: Daten aus der Collection lesen

APEX Collections basieren auf zwei „klassischen“ Datenbanktabellen. In `WWV_FLOW_COLLECTIONS$` sind die Benutzer-Sessions gespeichert, und in `WWV_FLOW_COLLECTION_MEMBERS$` leben die Daten, die den jeweiligen Benutzer-Session gehören und für andere Sessions nicht sichtbar sind. Diese beiden Tabellen sind in einer View namens `APEX_COLLECTIONS` gekapselt. Die „Magie“ liegt nun darin, dass der Programmierer sich lediglich auf diese View beziehen muss, wenn er die privaten Daten des Benutzers anzeigen möchte. Jegliches mühselige Session-Handling wird ihm von APEX abgenommen!

Somit ergibt das SQL-Statement

```
SELECT *
  FROM apex_collections
 WHERE collection_name = 'meine_collection'
```

tatsächlich nur die Datensätze, die dem jeweiligen Benutzer zugeordnet sind, ohne dass man sich um weitere Details wie Session-IDs, Usernamen-Zuordnung etc. kümmern müsste. Dies funktioniert allerdings nur, solange das SQL im APEX-Kontext abgesetzt wird und die Collection bereits angelegt und mit Daten gefüttert wurde. APEX kümmert sich im Hintergrund darum, dass die „richtigen" Zeilen in `APEX_COLLECTIONS` zu jeder Zeit auf jeder Anwendungsseite zur Verfügung stehen.

Als verantwortungsvoller Programmierer wollen Sie natürlich nicht `SELECT *` verwenden. Die Spalten, die beim Anlegen der Collection (s.o.) im SQL-Statement referenziert wurden, heißen in Collection (in der Reihenfolge ihres Auftretens) `C001, C002, …` bis `C050`. Sie sind vom Spaltentyp `varchar2(4000)`. Bei Bedarf können weitere fünf `number-` und fünf `date-`Spalten definiert werden, sowie jeweils eine Spalte vom Typ `CLOB`, `BLOB` und `XMLTYPE` (zu diesem Zweck gibt es überladene Varianten von `create_collection_from_query`).

Was ist nun der Primary Key jedes Datensatzes? Nun, einerseits wollen Sie vermutlich den technischen oder fachlichen PK der zugrundeliegenden Tabelle mit in die Collection-Daten schreiben (als Teil Ihrer SQL-Abfrage), andererseits ordnet APEX beim Initialisieren der Collection jedem Datensatz eine sogenannte Sequence-ID zu, die anhand der Reihenfolge in der Ergebnismenge in Einerschritten inkrementiert wird. Der entsprechende Spaltenname lautet `seq_id`; diese Spalte ist zusammen mit `collection_name` indiziert (alle übrigen Spalten der View `apex_collections` sind nicht indiziert).

Damit sieht der verbesserte Prototyp einer Collection-Abfrage so aus:

```
SELECT seq_id, c001, c002 --  und weitere Spalten...
  FROM apex_collections
 WHERE collection_name = 'meine_collection'
```

Schritt 3: Daten einfügen, ändern oder löschen

Das direkte Manipulieren der View `APEX_COLLECTIONS` per `INSERT, UPDATE` oder `DELETE` ist leider nicht vorgesehen (Abhilfe kann man jedoch mit Hilfe von `INSTEAD OF`-Triggern schaffen, über Views, die wiederum APEX_COLLECTIONS kapseln – dazu mehr im Vortrag). Anstatt direktem DML verwendet man Befehle aus der Package `APEX_COLLECTION`. Zum Verständnis der drei folgenden Methoden sollte man wissen, dass APEX die einzelnen Datensätze einer Collection als „Member" und die Spalten als „Attribute" bezeichnet und dass die Parameter `p_c001` bis `p_c050` wahlfrei sind (`default NULL`)

Einfügen:

```
new_seq := apex_collection.add_member
               (p_collection_name, p_c001, p_c002, p_c003, ...);
```

Ändern (vollständiger Datensatz):

```
apex_collection.update_member
               (p_collection_name, p_seq, p_c001, p_c002, p_c003, ...);
```

Ändern (einzelne Spalte):

```
apex_collection.update_member_attribute
               (p_collection_name, p_seq, p_attr_number, p_attr_value);
```

Löschen:

```
apex_collection.delete_member (p_collection_name, p_seq);
```

Es existieren weitere Befehle im Package `APEX_COLLECTION;` mit ihnen kann man beispielsweise…

- abfragen, ob eine Collection bereits existiert,
- eine leere Collection ohne SQL-Abfrage anlegen,
- mehrere Member gleichzeitig (als Array) mit einem Befehl einfügen,
- die Anzahl der Member herausfinden,
- die Reihenfolge der Member verändern bzw. nach bestimmten Spalten sortieren,
- prüfen, ob der Inhalt der Collection verändert wurde,
- für jeden Member einen MD5-Hash berechnen lassen und abspeichern,
- alle Member einer Collection löschen,
- eine bestimmte Collection löschen,
- alle Collections des Users löschen.

Auflösen der Collection-Daten gegen die fachlichen Tabellen

Hierin liegt bei weitem der aufwändigste Part. Es besteht nämlich keine automatische Beziehung zwischen den Daten, die man in die Collection geladen hat, und den Tabellen, in die man das geänderte Set zurückspeichern möchte. Typischerweise durchläuft man nach dem SUBMIT-Befehl jeden Datensatz der Collection, um zu prüfen, ob dieser neu ist oder geändert wurde. Gelöschte Datensätze sind, MRU-typisch, anhand eines Lösch-Kennzeichens zu erkennen, das beispielsweise als Checkbox-Spalte in die Collection integriert wird (hieraus ergibt sich übrigens die Chance, „gelöschte“ Datensätze beliebig lang aufzubewahren). Anschließend führt man zeilenweises SQL gegen die Originaltabellen aus.

Darüber hinaus ist die Anwendung von Locking-Mechanismen nicht trivial, da der von APEX verwendete Automatismus bei manuellen MRUs nicht greift. Man kann stattdessen die MD5-Checksumme heranziehen oder mit Versionsnummern auf Datensätzen arbeiten.

Spätestens an dieser Stelle kommt der Wunsch nach einer Hilfs-Package auf, welche die Programmierung rund um APEX Collections vereinfacht. Viel Arbeit lässt sich außerdem mit einem Code-Generator sparen, den man von Projekt zu Projekt weiter ausbauen kann. Diesen stelle ich Ihnen in meiner Präsentation vor.

Ausblick

Zusammenfassend lässt sich sagen, dass erfreulich wenig Hindernisse existieren, um grundsätzlich erfolgreich mit APEX Collections zu arbeiten. Zwei unschätzbare Vorteile von Collections gegenüber herkömmlichen tabellarischen MRUs, die besonders in größeren Projekten zum Tragen kommen, seinen hier nochmals in Erinnerung gebracht:

1. APEX Collections sind „private Tabellen“; der Benutzer kann dort Daten speichern, ohne die Original-Datenbanktabellen zu beeinflussen
2. Collection-Daten sind persistent in allen Anwendungen verfügbar, die mit derselben APEX Session-ID geöffnet werden

In meinem Vortrag am 14.06 auf der APEX 2012 Development-Konferenz gebe ich viele (hauptsächlich) positive Erfahrungen an Sie weiter, die wir in einem großen Projekt mit APEX Collections gesammelt haben. Der Fokus meiner Präsentation liegt auf der Demonstration weiterer Features, die mit den Werkzeugen der APEX COLLECTION API (und etwas Hilfe von jQuery und CSS) möglich sind, um Formulare zu erstellen, die das Attribut „de luxe“ verdient haben.

Kontaktadresse:

Andreas Wismann
WHEN OTHERS
Hirschstraße 10
D-41564 Kaarst

Telefon: +49 (0) 2131 - 314 9966
Fax: +49 (0) 2131 – 314 9967
E-Mail wismann@when-others.com
Internet: http://when-others.com