# So geht Fuzzy-Suche in Application Express

Carsten Czarski, ORACLE Deutschland B.V. & Co KG

„Suche“ ist allgegenwärtig – auch in Unternehmensanwendungen. Ob es um Kunden, Lieferanten, Rechnungen, Aufträge oder etwas anderes geht: Der Anwendungsfall, etwas suchen zu müssen, ist stets vorhanden. So liegt diese Anforderung oft auch auf dem Schreibtisch des Apex-Entwicklers.

Die gute Nachricht ist: Da Apex in der Datenbank läuft, stehen dem Entwickler alle Funktionen zur Verfügung und mit Oracle Text enthält die Datenbank eine Volltextsuche, die linguistische Funktionen und Ähnlichkeitssuche unterstützt. Darüber hinaus kann man mit ein wenig Programmierung auch normale Tabellendaten mit Oracle Text indizieren und so für die Ähnlichkeitssuche verfügbar machen.

In einer Apex-Anwendung werden Daten typischerweise mit einem interaktiven oder klassischen Bericht visualisiert; seit Apex 5.1 kommt noch das Interactive Grid hinzu. Interactive Grid und der interaktive Bericht bringen das Suchfeld schon mit – man muss als Entwickler also gar nichts mehr tun *(siehe Abbildung 1)*.

Das Ganze funktioniert in den meisten Fällen ganz wunderbar, hat aber Grenzen:

- Sind die Datenbestände sehr groß, leidet bei freier Suche schnell die Performance.
- Man braucht Indizes auf jeder durchsuchbaren Tabellenspalte.
- Die Suche ist nicht fehlertolerant: So lassen sich fehlerhaft gespeicherte Daten oft nur schwer finden. Besonders Namen machen den Anwendern häufig das Leben schwer; fremdsprachige Sonderzeichen und Schreibweisen sorgen oft dafür, dass diese Namen nur schwer auffindbar sind.

Teilweise können diese Anforderungen mit SQL-Funktionen umgesetzt werden; so bietet die Datenbank an, diakritische Zeichen bei SQL-Abfragen zu ignorieren. Die SQL-Funktion „SOUNDEX“ stellt eine Ähnlichkeitssuche zur Verfügung, funktioniert aber nur für die englische Sprache. Eine generische Ähnlichkeitssuche, die für mehrere Sprachen funktioniert, lässt sich nur mit SQL jedoch nicht umsetzen.

Allerdings gibt es seit geraumer Zeit Oracle Text in allen Datenbank-Editionen. Ursprünglich zur Suche in Dokumentbeständen konzipiert, kann es, mit ein wenig PL/SQL-Programmierung, auch für normale Tabellendaten genutzt werden. Mit seiner Hilfe lassen sich linguistische Funktionen oder die Ähnlichkeitssuche auch für Tabellendaten nutzen.

Hinweis: Die Code-Beispiele in diesem Artikel basieren auf einem Blog-Posting

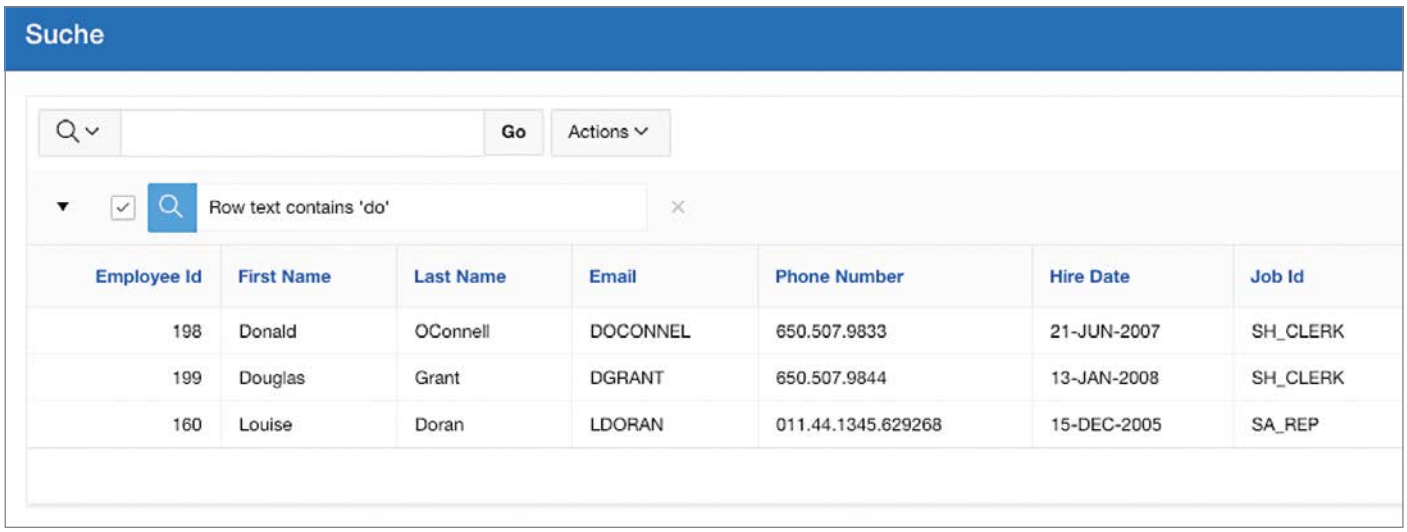


*Abbildung 1: Interaktiver Bericht in Application Express*

```
select employee_id, first_name, last_name
  from employees_search
 where contains( full_name, 'mustermann and max' ) > 0
```

*Listing 1: Eine typische Oracle-Text-Suche als SQL-Abfrage*

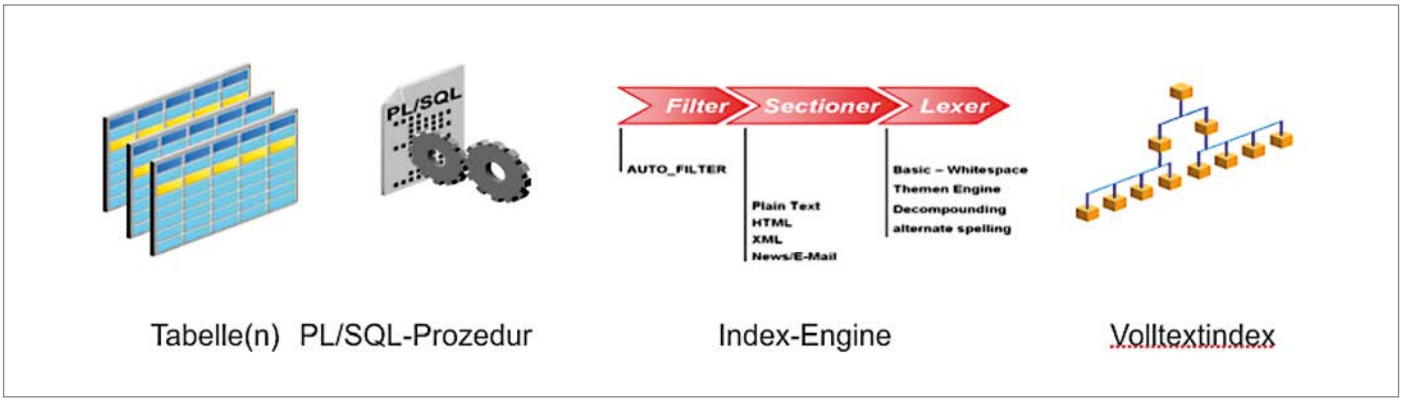


*Abbildung 2: Eine PL/SQL-Prozedur stellt die zu indizierenden Daten zusammen*

des Autors und sind hier aus Platzgründen nicht vollständig wiedergegeben. Im Blog-Posting *(siehe „http://oracle-text-de.blogspot.de/2012/03/nochmal-userdatastore-ein-umfassendes.html“)* finden sich die vollständigen Skripte, die auf dem Oracle-Beispielschema „HR“ basieren und in der eigenen Umgebung eingespielt werden können.

## Oracle Text

Ein Oracle-Text-Index funktioniert anders als normale Indizes. Zwar wird auch dieser mit dem Kommando „CREATE INDEX“ erzeugt, damit erschöpfen sich die Gemeinsamkeiten allerdings auch schon. Oracle Text fasst die zu indizierenden Daten als Dokumente auf. Diese liegen entweder direkt in der Tabelle oder sie werden während der Indizierung generiert. Dabei sind die Dokumente zunächst in Abschnitte („Sections“) und dann in Tokens zerlegt. Nach den Tokens kann dann gesucht werden. Jeder Schritt lässt sich einzeln konfigurieren und steuern.

Ist der Index erzeugt, wird eine Volltextsuche per SQL-Abfrage durchgeführt, wobei Oracle Text mit dem Suchoperator „CONTAINS“ angesprochen wird. *Listing 1* zeigt eine typische Oracle-Text-Suchabfrage.

Wichtig für die Erstellung des Oracle-Text-Index, ganz besonders im Kontext dieses Artikels, ist die „Datastore Preference“. Diese besagt, wo die Daten, die Oracle Text indizieren soll, herkommen.

Im Gegensatz zu einem normalen Index kann man mit Oracle Text eine Datenbank-Spalte indizieren – die konkreten Daten kommen jedoch ganz woanders her. Datenquellen können dann andere Spalten der gleichen Tabelle, andere Tabellen, das Dateisystem oder sogar entfernte Daten im Netzwerk sein. Die folgende Aufzählung zeigt einige Datastore-Varianten, die von Oracle Text bereitgestellt werden:

- *DIRECT_DATASTORE*
  Die Daten werden aus der Tabellen-Spalte entnommen, für die der Index

erzeugt wird. Das ist der Standard und auch das Verhalten, das man bei einer Index-Erstellung zunächst erwartet.

- *MULTICOLUMN_DATASTORE*
  Der Index wird für eine Tabellen-Spalte angegeben; tatsächlich werden allerdings die im Multicolumn-Datastore angegebenen Spalten indiziert.

- *FILE_DATASTORE*
  Die Tabellen-Spalte speichert nur Dateisystem-Pfade ab und Oracle Text holt sich die Daten aus dem Dateisystem des Datenbank-Servers.

- *USER_DATASTORE*
  Oracle Text ruft für jede Tabellenzeile eine PL/SQL-Prozedur auf und gibt das zu indizierende Dokument zurück. Die Implementierung ist völlig frei.

„USER_DATASTORE" bietet die meisten Freiheitsgrade: Im PL/SQL-Code können Daten beliebig aus Tabellen selektiert, nachbearbeitet und dann als Dokument zusammengestellt werden. Das generierte Dokument wird indiziert und danach verworfen (es muss also kein Plattenplatz dafür bereitgestellt werden). *Abbildung 2* zeigt den Zusammenhang schematisch.

Die PL/SQL-Prozedur muss eine von Oracle Text vorgegebene Signatur haben.

```
create or replace procedure datastore_procedure(
   rid        in             rowid,
   tlob       in out nocopy varchar2 | clob | blob )
```

*Listing 2: Signatur der PL/SQL-Prozedur für einen USER_DATASTORE*

```
begin
   ctx_ddl.create_preference(
       preference_name => 'my_datastore_preference',
       object_name     => user_datastore' );
   ctx_ddl.set_attribute(
       preference_name => 'my_datastore_preference',
       attribute_name  => 'procedure',
       attribute_value => 'datastore_procedure' );
end;
```

*Listing 3: Registrieren der PL/SQL-Prozedur als „USER_DATASTORE"*

```
create index ft_volltext on {tabelle}({spalte})
indextype is ctxsys.context
parameters('
   datastore      my_datastore_preference
   lexer          my_lexer_preference
   stoplist       ctxsys.empty_stoplist
   memory 500M')
```

*Listing 4: Erzeugen des Volltext-Index*

An dieser ist bereits erkennbar, wie sie von Oracle Text während der Indizierung verwendet wird: Für jede indizierte Zeile wird sie mit der „ROWID" aufgerufen und das zu indizierende Dokument als „OUT"-Parameter zurückgegeben *(siehe Listing 2)*. Die Prozedur kann „VARCHAR2", „CLOB" oder sogar binäre Daten als „BLOB" zurückgeben.

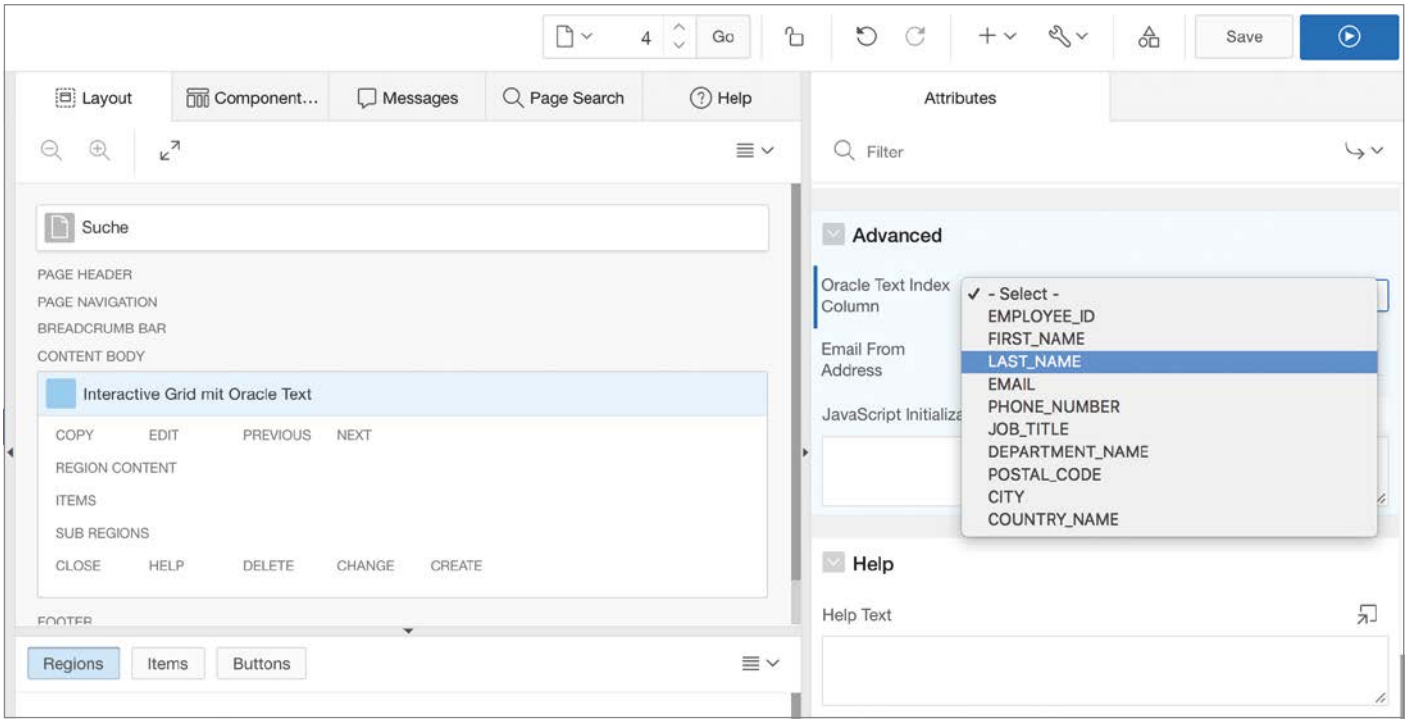


*Abbildung 3: Deklarative Nutzung von Oracle Text im Interactive Grid (ab Apex 5.1)*

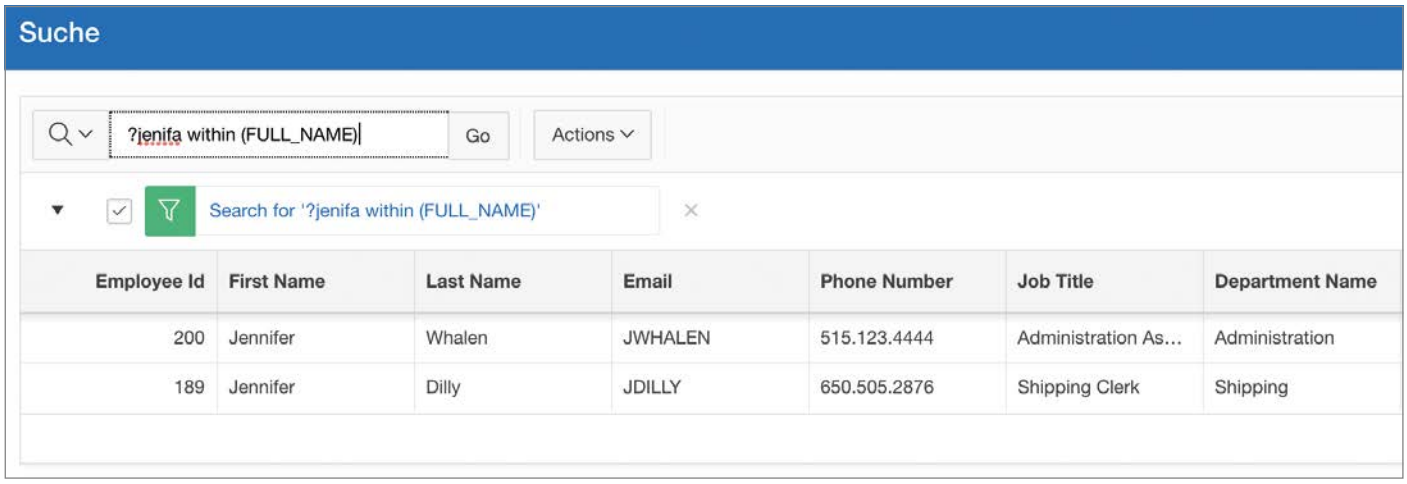


*Abbildung 4: Volltextsuche im Interactive Grid*

```
create or replace function convert_to_text_query(
   p_end_user_query in varchar2
) return varchar2 is
begin
   return '?' || p_end_user_query || ' within (FULL_NAME)';
end convert_to_text_query;
```

*Listing 5: PL/SQL-Funktion zur Umwandlung einer Abfrage in Oracle-Text-Syntax*

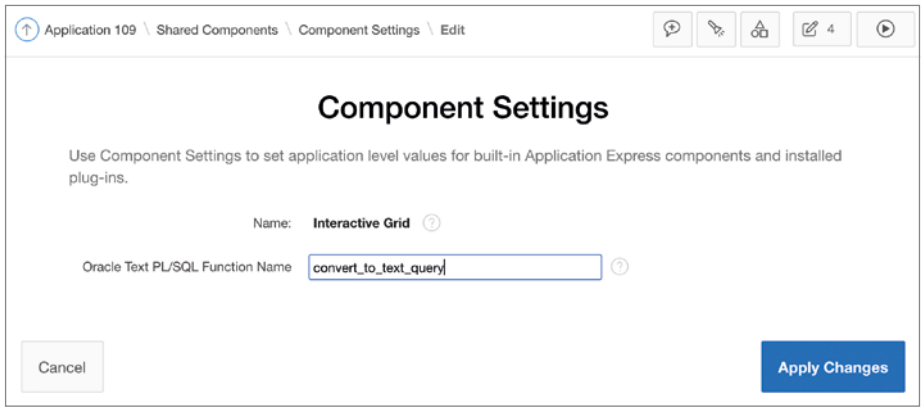


*Abbildung 5: PL/SQL-Funktion als Component Setting für das Interactive Grid*

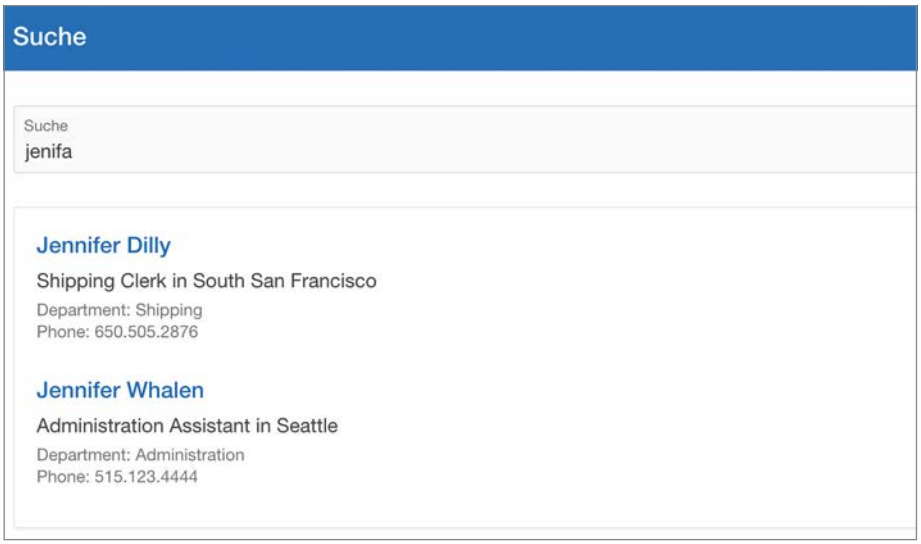


*Abbildung 6: Einfache Such-Anfragen reichen aus – Oracle-Text-Syntax ist nicht nötig*

Wenn bereits im Vorfeld sicher ist, dass die Größe der generierten Dokumente 32.767 Byte nicht übersteigt, empfiehlt sich die Nutzung von „VARCHAR2" als Datentyp für den „OUT"-Parameter. Im nächsten Schritt wird die PL/SQL-Prozedur für Oracle Text registriert, indem man eine Datastore-Preference erzeugt *(siehe Listing 3)*.

Durch Preferences können sehr viele Eigenschaften des Volltext-Index konfiguriert werden; eine vollständige Aufzählung würde den Rahmen dieses Artikels sprengen. Beispiele finden sich im eingangs erwähnten Blog-Posting. Sind alle Preferences erzeugt, wird im letzten Schritt der Index angelegt *(siehe Listing 4)*.

## Oracle Text in Apex nutzen: Interactive Grid

Die einfachste Möglichkeit, einen Oracle-Text-Index in einer Apex-Anwendung zu nutzen, ist das ab Version 5.1 verfügbare Interactive Grid. Im Page Designer ist lediglich die mit Oracle Text indizierte Tabellenspalte als „Oracle Text Index Column" auszuwählen *(siehe Abbildung 3)*. Endbenutzer verwenden nun einfach das Suchfeld oberhalb des Interactive Grid. Hier kommen nun die Oracle-Text-Filtersyntax und damit die linguistischen Funktionen zum Einsatz *(siehe Abbildung 4)*.

Allerdings ist die Abfrage-Syntax von Oracle Text sehr technisch und für einen Endbenutzer nur schwer verständlich. Oft ist es auch nicht gewünscht, dem Endbenutzer alle Suchfunktionen verfügbar zu machen. Es braucht also die Übersetzung einer Suchanfrage des Endbenutzers in die Oracle-Text-Syntax. Dazu dient eine PL/SQL-Funktion; der Code in *Listing 5*

dient als bewusst vereinfachtes Beispiel. Die PL/SQL-Funktion ist in Apex in den „Component Settings“ des Interactive Grid hinterlegt. Diese sind in den „Shared Components“ zu finden *(siehe Abbildung 5)*.

Von nun an reicht ein einfacher Suchbegriff aus: Die PL/SQL-Funktion wird von Apex automatisch aufgerufen und es findet tatsächlich eine Abfrage mit Oracle-Text-Syntax statt. So lässt sich eine eigene Abfragesprache implementieren, die von Oracle Text komplett abstrahiert *(siehe Abbildung 6)*.

Interactive Grid ist somit eine sehr einfache Möglichkeit, Oracle Text in Apex zu nutzen. Allerdings ist die Grid-Oberfläche für eine Suchfunktion oft ungeeignet. Ein Look & Feel, wie es von Suchmaschinen angeboten wird, lässt sich so nicht realisieren. Nimmt man andere Apex-Komponenten zu Hilfe, ist das aber überhaupt kein Problem.

## Oracle Text in Apex nutzen: Einfache Berichte

Prinzipiell kann Oracle Text in jeder Apex-Komponente verwendet werden, denn alle Apex-Komponenten basieren auf SQL-Abfragen als Datenquelle. Ein klassischer Bericht und ein Textfeld für den Suchbegriff dienen als Basis für die „Apex-Suchmaschi-

```
select employee_id, first_name, last_name
  from employees_search
 where contains(
         last_name,
         convert_to_text_query(:P5_SUCHE)) >0
       and :P5_SUCHE is not null
```

*Listing 6: Oracle Text SQL-Abfrage als Basis für den Bericht*


*Abbildung 7: Klassischer Bericht mit Oracle Text in Aktion*

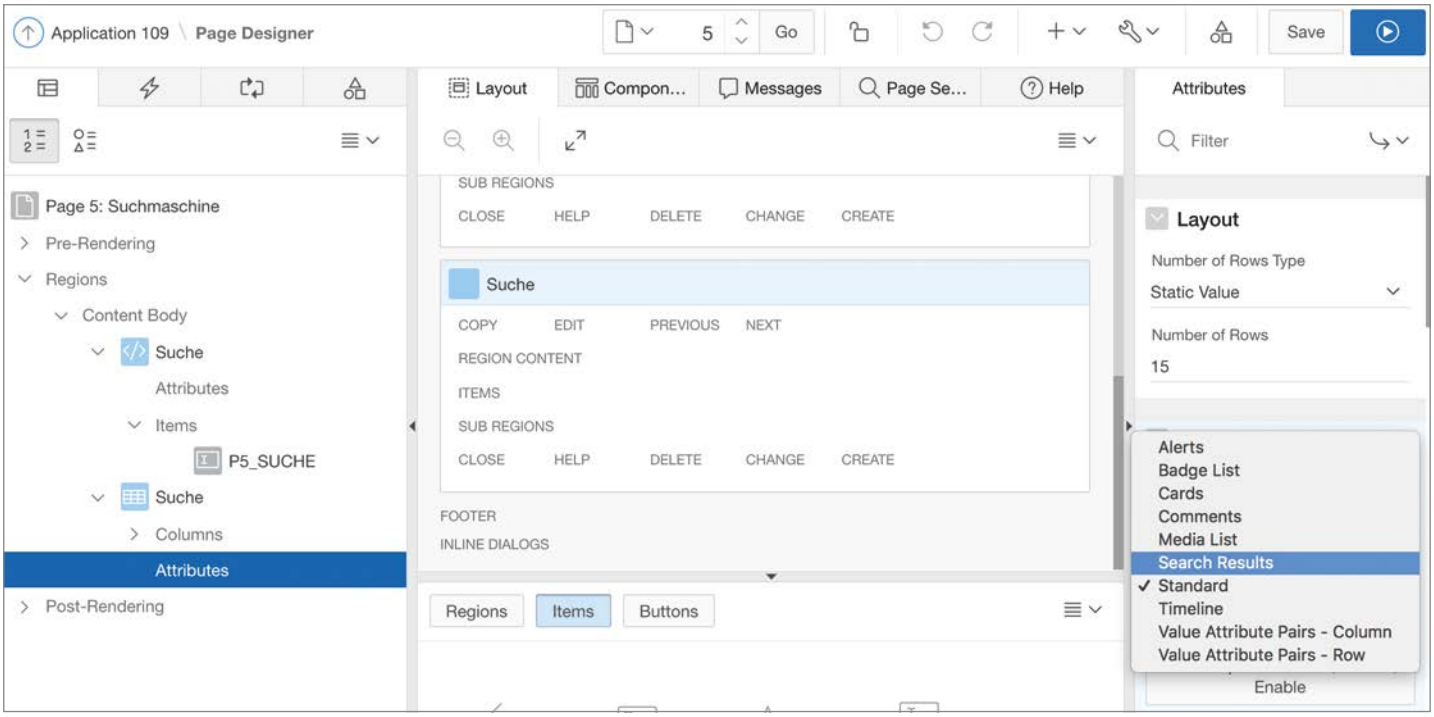


*Abbildung 8: Neues Berichts-Template auswählen: „Search Results“*

```
select first_name || ' ' || last_name as search_title,
       job_title || ' in ' || city      as search_desc,
       'Department'                     as label_01,
       department_name                  as value_01,
       'Phone'                          as label_02,
       phone_number                     as value_02,
       '{target-link}'.                 as search_link
  from employees_search
where contains (...) > 0
```

*Listing 7: Für das „Search Results"-Template angepasste SQL-Abfrage*

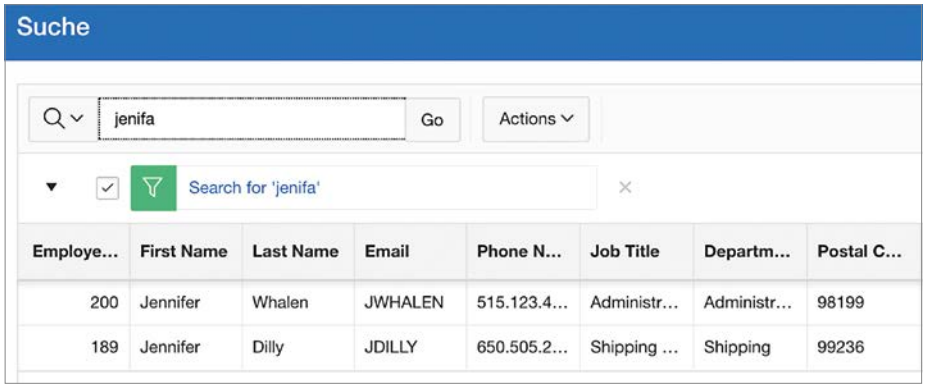


*Abbildung 9: Suchmaschine in Apex*

ne". *Listing 6* zeigt die SQL-Abfrage für den Bericht; die bereits erwähnte PL/SQL-Funktion zur Übersetzung der Suchanfrage in die Oracle-Text-Syntax wird gleich mitverwendet. Die Eingabe des Benutzers wird als Bindevariable „:P5_SUCHE" angesprochen. Das vorläufige Ergebnis sieht dann in etwa so wie in *Abbildung 7* aus.

Das sieht allerdings immer noch nicht nach einer Suchmaschine aus. Standardmäßig wird auch ein klassischer Bericht in Tabellenform dargestellt. Jedoch lässt sich das Aussehen mit „Report Templates" beliebig gestalten. Zur Darstellung als Ergebnisliste einer Suchmaschine bietet Apex bereits ein fertiges Template namens „Search Results", das man nur in den Berichts-Attributen auswählen muss *(siehe Abbildung 8)*.

Das Berichts-Template „Search Results" erwartet, dass die SQL-Abfrage ganz bestimmte Spalten zurückliefert – also ist das SQL ein wenig zu ändern *(siehe Listing 7)*. Die neue Apex-Seite sieht dann schon eher nach einer Suchmaschine aus.

Die Ergebnisspalte „SEARCH_LINK" kann einen Link zu einer Detailseite enthalten. Damit sind nahezu alle Daten für eine Suche verfügbar und der Endbenutzer findet in der Unternehmensanwendung das vor, was er aus dem Internet bereits kennt.

## Fazit und Ausblick

Da Apex in der Oracle-Datenbank läuft, liegt es nahe, die vorhandenen Datenbank-Funktionen so gut es geht auszunutzen. Oracle Text ist in allen Datenbank-Editionen und -Versionen enthalten; somit ist es naheliegend, den Volltext-Index in der Apex-Anwendung praktisch einzusetzen. Mit ein wenig PL/SQL-Programmierung lässt sich ein Oracle-Text-Index für normale Tabellendaten erzeugen. Der geschickte Einsatz von Apex-Komponenten und ein wenig Nacharbeit am Look & Feel sorgen dafür, dass eine Suchmaschine für Tabellendaten sehr schnell bereitsteht.

Ein eigenes Thema ist das Verhalten des Oracle-Text-Index, wenn sich die Daten in den zugrunde liegenden Tabellen ändern. Je nach Anforderung ist dies entweder ein einfacher Scheduler-Job, der den Text-Index regelmäßig aktualisiert, oder es muss noch etwas Zeit in PL/SQL-Programmierung und Oracle-Text-Konfiguration investiert werden.

## Weitere Informationen

- Oracle Application Express im OTN: *http://www.oracle.com/technetwork/developer-tools/apex/overview/index.html*
- Oracle Application Express Demoserver: *http://apex.oracle.com*
- Blog Posting: *http://oracle-text-de.blogspot.de/2012/03/nochmal-userdatastore-ein-umfassendes.html*

Carsten Czarski
carsten.czarski@oracle.com

# Weniger als 15 Prozent der „On-Prem Application"-Kunden haben bislang in die Cloud gewechselt

Oracle hat im März 2018 die Finanzzahlen für das dritte Quartal des Geschäftsjahres bekanntgegeben. Der Gesamtumsatz stieg im Vergleich mit dem Vorjahreszeitraum um 6 Prozent auf 9,8 Milliarden US-Dollar. Nach Aussage von Oracle-CEO Mark Hurd nähert sich das eigene „Cloud SaaS Applications"-Geschäft der Grenze von 5 Milliarden US-Dollar. Allerdings hätten weniger als 15 Prozent der eigenen „On-Premise Application"-Kunden bislang ihre Anwendungen in die Cloud verlegt. Für die anderen 85 Prozent sieht er einen chancenreichen Markt.