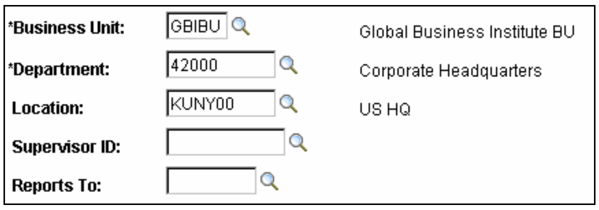
PeopleSoft: Suppressing Automatic Lookups for Improved Performance and Usability

by David Kurtz, Go-Faster Consultancy Ltd.

One of the catchphrases that I have adopted[1] is 'the fastest way to do something is not to do it at all'. Instead of making something take less time by tuning, it is much better make it take no time at all by not doing it in the first place!

This applies to PeopleSoft as much as anything else. In this article I want to draw attention to the Lookup exclusion table, a very valuable but almost unknown usability feature that was only recently brought to my attention.

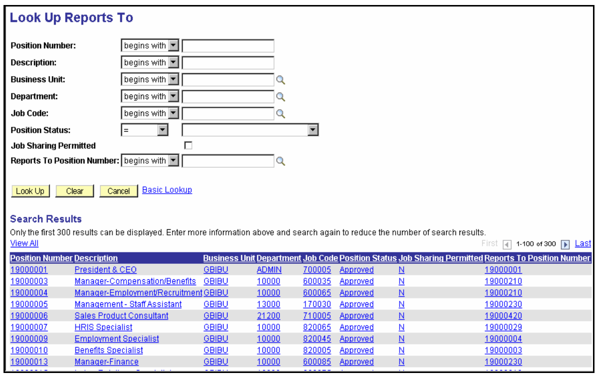
Lots of fields in PeopleSoft applications have lookups associated with them, indicated by the magnifying glass icon (see figure 1).



*Business Unit:	GBIBU	Global Business Institute BU
*Department:	42000	Corporate Headquarters
Location:	KUNY00	US HQ
Supervisor ID:		
Reports To:		

Figure 1. Screenshot from PeopleSoft Application

When you click on that icon you are taken to a lookup dialogue, but by default the search fires automatically. Sometimes the operator must wait a while for the SQL queries to return the data. However, if there is a lot of data only the first 300 rows are retrieved into the component buffer, and only the first 100 rows of that set are shown on the first page. Figure 2 shows Applicant Hire in HR from the demo database as an example.



Look Up Reports To

Position Number:	begins with	
Description:	begins with	
Business Unit:	begins with	
Department:	begins with	
Job Code:	begins with	
Position Status:	=	
Job Sharing Permitted		☐
Reports To Position Number:	begins with	

Search Results

Only the first 300 results can be displayed. Enter more information above and search again to reduce the number of search results.

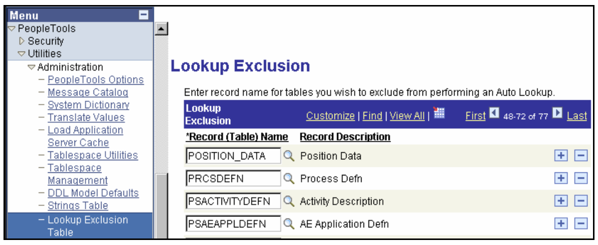
View All — First 1-100 of 300 Last

Position Number	Description	Business Unit	Department	Job Code	Position Status	Job Sharing Permitted	Reports To Position Number
19000001	President & CEO	GBIBU	ADMIN	700005	Approved	N	19000001
19000003	Manager-Compensation/Benefits	GBIBU	10000	600035	Approved	N	19000210
19000004	Manager-Employment/Recruitment	GBIBU	10000	600065	Approved	N	19000210
19000005	Management - Staff Assistant	GBIBU	13000	170030	Approved	N	19000230
19000006	Sales Product Consultant	GBIBU	21200	710005	Approved	N	19000420
19000007	HRIS Specialist	GBIBU	10000	820065	Approved	N	19000029
19000009	Employment Specialist	GBIBU	10000	820045	Approved	N	19000004
19000010	Benefits Specialist	GBIBU	10000	820005	Approved	N	19000003
19000013	Manager-Finance	GBIBU	10000	600085	Approved	N	19000230

Figure 2. Look-up results

These results are almost certainly useless to the operator who must then enter criteria into the lookup search dialogue and search again. It would be better if the first automatic search could be suppressed. From PeopleTools 8.44, the prompt table associated with the field can be put onto the lookup exclusion table (as shown in figure 3).

[1]I think from Cary Millsap of Hotsos (www.hotsos.com).



Lookup Exclusion

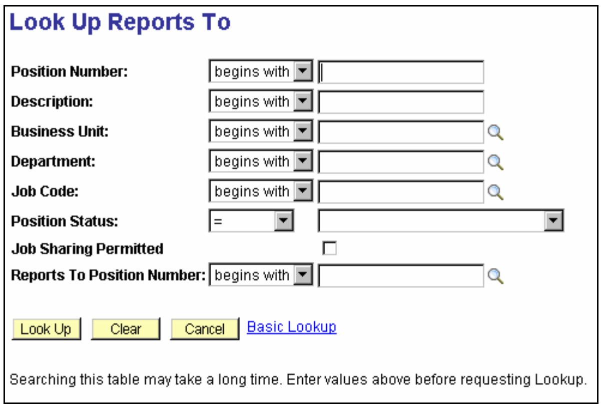
Enter record name for tables you wish to exclude from performing an Auto Lookup.

Lookup Exclusion | Customize | Find | View All | First 48-72 of 77 Last

*Record (Table) Name	Record Description
POSITION_DATA	Position Data
PRCSDEFN	Process Defn
PSACTIVITYDEFN	Activity Description
PSAEAPPLDEFN	AE Application Defn

Figure 3. Lookup exclusion table maintenance page

Now the user goes straight into a page where they are prompted to enter search criteria to find the relevant data. The user doesn't have to enter any criteria before pressing the Look Up button, but there is a message warning them that if they don't if may take a long time. The benefit to the user is immediate. You don't have to restart any application servers before the behaviour of the lookup to change.


Look Up Reports To

Position Number:	begins with	
Description:	begins with	
Business Unit:	begins with	
Department:	begins with	
Job Code:	begins with	
Position Status:	=	
Job Sharing Permitted	☐	
Reports To Position Number:	begins with	

Searching this table may take a long time. Enter values above before requesting Lookup.

Figure 4. Search criteria on excluded lookup

The lookup exclusion table is a very simple and quick way of improving performance by cutting out unnecessary work. It is easier to implement these changes because there are only configuration, rather than customisation.

Identifying Candidate Records for the Lookup Exclusion Table

Although an end user could identify which look-ups to suppress, working out which prompt table is behind a particular field is a job for a developer.

The following query will identify all the records that are used as look-up records within a PeopleSoft system.

```
SELECT DISTINCT r.recname
FROM pspnlfield p
,    psrecfielddb f
,    psrecdefn r
WHERE p.fieldname = f.fieldname
AND p.recname = f.recname
AND r.recname = f.edittable
```

When I first looked at the feature, I took the view that this is a performance issue, and took a minimalist approach. I would only suppress lookups that were consuming a significant amount of time. However, there appear to be no negative effects of having a large number records in the exclusion table because this information is cached by the application server.

The next question is: what is the criteria for adding a record to the lookup exclusion table?

Certainly if a lookup record returns more than 300 rows it should not be executed automatically, because only the first 300 rows will be returned. The user will probably not have the row they are looking for in the result set and they will have to add search criteria and repeat the search.

There is a good case for suppressing the automatic lookup when it returns more than 100 rows. The results will return up to 3 pages of 100 rows per page. Then the user may well have to navigate to a different page in order to find what they are looking for.

Having discussed this issue with a few people, there is some consensus that the automatic search should be suppressed for records with more than 25-50 rows. If there are more rows than this, the user is likely to have to scroll down the page the desired row.

The number of rows that a lookup record may retrieve is therefore a good criteria for deciding whether to suppress the automatic lookup. So I have constructed a SQL script to identify all candidate records and count the number of rows that each returns. First I need a table in which to store the results.

```
CREATE TABLE lux_cand
(recname       VARCHAR2(15) NOT NULL
,sqltablename  VARCHAR2(18)
,rectype       NUMBER NOT NULL
,num_lookups   NUMBER NOT NULL
,num_rows      NUMBER
,lead_key      VARCHAR2(18)
,num_lead_keys NUMBER
,lastupddttm   DATE )
;
CREATE UNIQUE INDEX lux_cand ON
lux_cand(recname)
;
```

This table will be populated with all the lookup exclusion fields in the system.

```
INSERT INTO lux_cand (recname,
sqltablename, rectype, lead_key,
num_lookups)
SELECT r.recname
,      CASE WHEN r.rectype IN(0,1,6)
THEN /*regular records*/
        DECODE(r.sqltablename,'
','PS_'r.recname,r.sqltablename)
       END
,      r.rectype
,      k.fieldname
,      count(*) num_lookups
FROM   pspnlfield p
,      psrecfielddb f
,      psrecdefn r
LEFT OUTER JOIN pskeydefn k
  ON k.recname = r.recname
  AND k.indexid = '_'
  AND k.keyposn = 1
WHERE  p.fieldname = f.fieldname
AND    p.recname = f.recname
AND    f.edittable > ' '
AND    r.recname = f.edittable
GROUP BY r.recname, r.rectype,
k.fieldname
,     DECODE(r.sqltablename,'
','PS_'r.recname,r.sqltablename);
```

Lookup tables that are keyed on SETID only return rows for that SETID, so we need to identify them and process them differently.

```
UPDATE lux_cand l
SET    l.lead_key = 'SETID'
WHERE  l.lead_key IS NULL
AND    EXISTS(
  SELECT 'x'
  FROM psrecfielddb k
  WHERE k.recname = l.recname
  AND k.fieldname = 'SETID'
  AND MOD(k.useedit,2) = 1 /*key*/)
;
```

Otherwise I what to know what is the leading column of key index.

```
UPDATE lux_cand l
SET    l.lead_key = (
  SELECT k.fieldname
  FROM psrecfielddb k
  WHERE k.recname = l.recname
  AND MOD(k.useedit,2) = 1
  AND k.fieldnum = (
    SELECT MIN(k1.fieldnum)
    FROM psrecfielddb k1
    WHERE k1.recname = k.recname
    AND MOD(k1.useedit,2) = 1 ))
WHERE l.lead_key IS NULL
;
```

Tables and views that do not exist in the database are marked.

```
UPDATE lux_cand l
SET    l.sqltablename = ''
WHERE  l.sqltablename IS NOT NULL
AND    l.rectype = 0 /*table*/
AND    NOT EXISTS(
  SELECT 'x'
  FROM user_tables o
  WHERE o.table_name = l.sqltablename)
;
UPDATE lux_cand l
SET   l.sqltablename = ''
WHERE l.sqltablename IS NOT NULL
AND   l.rectype IN(1,6) /*views*/
AND NOT EXISTS(
  SELECT 'x'
  FROM user_views o
  WHERE o.view_name = l.sqltablename);
```

I have identified some views for which it takes a very long time to count the rows. These should definitely be on the lookup exclusion table regardless. You may need to add to this list if the subsequent step of the test hangs on any other view.

```
UPDATE lux_cand l
SET l.lastupddttm = SYSDATE,
l.num_rows = 999
WHERE (l.recname IN('PSFIELDRECDVW',
'DEPARTMENT_SRCH', 'DEPT_TBL_ACCESS',
'ER_DEPT_TBL_VW', 'JOB_REQ_OP_SRCH',
'PAY_EMPSRCH_NLD', 'PCMP_COMPAUTHVW',
'TC_EE_SRCH1', 'TL_ADM_NAME_VW',
'TL_P_EMPLID_VW', 'TM_TREE_EFDT_VW',
'TREE2_VW', 'TRN_DEPTEFFD_VW')
  OR l.recname LIKE 'EMPLMT_SRCH%'
  OR l.recname LIKE 'PERS_SRCH%')
AND l.lastupddttm IS NULL;
```

Now, I can count the number of rows in each table or view. For views keyed on SETID, I want to know the maximum number of rows for any one SETID.

```
DECLARE
l_num_rows INTEGER;
l_lead_keys INTEGER;
BEGIN
  FOR l IN (
    SELECT l.* FROM lux_cand l
    WHERE l.lastupddttm IS NULL
    AND l.sqltablename IS NOT NULL
    ORDER BY SIGN(l.rectype) /*do
tables first*/
    , CASE WHEN 'PS_'l.recname =
l.sqltablename THEN 0 ELSE 1 end
/*tools tables last*/
   , l.rectype, l.recname
  ) LOOP
    l_num_rows := 0;
    l_lead_keys := 0;
    IF l.lead_key = 'SETID' THEN
/*count max rows per setid*/
      EXECUTE IMMEDIATE 'SELECT
MAX(num_rows), COUNT(*) FROM (SELECT
COUNT(*) num_rows FROM
'l.sqltablename' GROUP BY
'l.lead_key')' INTO l_num_rows,
l_lead_keys;
      UPDATE lux_cand
      SET num_rows = NVL(l_num_rows,0)
      , num_lead_keys =
                     NVL(l_lead_keys,0)
      , lastupddttm = SYSDATE
      WHERE recname = l.recname;
    ELSE /*count number of rows*/
      EXECUTE IMMEDIATE 'SELECT
COUNT(*) FROM 'l.sqltablename' WHERE
rownum <= 301' INTO l_num_rows;
      UPDATE lux_cand
      SET num_rows = NVL(l_num_rows,0)
      ,lastupddttm = SYSDATE
      WHERE recname = l.recname;
    END IF;
    COMMIT;
  END LOOP;
```

```
END;
/
```

Now, the table LUX_CAND contains a list of all the tables and views used as lookups, and the number of rows that they return (I cannot do anything about PeopleSoft dynamic views). The next thing is to add any record that returns too many rows into the lookup exclusion table.

```
INSERT INTO psrecxl
SELECT recname FROM lux_cand
WHERE num_rows > 50
MINUS SELECT recname FROM psrecxl
/
```

The application server caches the list of excluded tables. There is no version number on PSRECXL. Instead, when you save an update to the lookup exclusion table, there is Component PeopleCode that updates the SYS and PPC version numbers, which causes the application server to refresh its cache for table, so that changes take effect immediately. So when updating PSRECXL with this script, it is also necessary to increment the version numbers.

```
UPDATE PSVERSION
SET VERSION = VERSION + 1
WHERE OBJECTTYPENAME IN('SYS','PPC')
;
UPDATE PSLOCK
SET VERSION = VERSION + 1
WHERE OBJECTTYPENAME IN('SYS','PPC')
;
COMMIT
;
```

On a Financials 8.4 demo system I found 4089 records being used as lookup records. Of those 557 have over 300 rows, 1058 have over 100 rows, and 1711 records return over 25 rows. On HR, I have suppressed the look up on 1711 records.

This script gives you a good first approximation of which lookups should be suppressed. If the users complain that you have either missed one, or suppressed one that you shouldn't, then you can adjust the list manually. Overall, suppressing these lookups will reduce user response time, thus improving their experience, while simultaneously saving system resources.

David Kurtz is a performance specialist working Enterprise PeopleSoft applications and Oracle RDBMS (www.go-faster.co.uk). He is the author of 'PeopleSoft for the Oracle DBA', published by Apress (www.psftdba.com). He is the chair of UKOUG's UNIX SIG, and has recently become a UKOUG Director with responsibility for representing the PeopleSoft community.