

EFFICIENT DATABASE PROGRAMMING WITH ABAP

ABAP152

Exercises / Solutions

**ULRICH KOCH, SAP AG,
TOBIAS WENNER, SAP AG**

ABAP152

The following four exercises fulfill the same task: They select all flights of carrier LH (Lufthansa) where the first class is fully booked (seatsmax_f = seatsocc_f). Then they retrieve all bookings for these flights and calculate the total revenue from these bookings (loccuram = local currency amount). Every exercise gets its result in a faster way than its predecessor until the final solution with exercise 4 is reached.

Exercise 1: Keep the Hit List Small

- Open program ZDABAP152_1_xx for editing (xx = the number of your group).
- Optimize the SELECT statements in form "version1": Substitute the CHECK-conditions by specifying the desired rows in a WHERE-clause.
- Run the program to see the effect of your optimization.

Solution 1: Keep the Hit List Small

```
*-----*
*  FORM VERSION1
*-----*
form version1.
* Version 1
  data: sflight_wa type sflight,
        sbook_wa type sbook,
        sum type sbook-loccuram value 0,
        t1 type i, t2 type i, delta(16) type p.

  get run time field t1.                                "start clock

* SELECT all columns, specific rows.
  select * from sflight into sflight_wa
    where carrid      = 'LH'
    and seatsmax_f = sflight~seatsocc_f.
  select * from sbook into sbook_wa
    where carrid = sflight_wa-carrid
    and connid = sflight_wa-connid
    and fldate = sflight_wa-fldate
    and loccurkey = 'EUR'.
    sum = sum + sbook_wa-loccuram.
  endselect.
endselect.

  get run time field t2.                                "stop clock
  delta = t2 - t1.
  skip.
  write: / 'SELECT all columns, specific rows:' color col_heading.
  write: / 'Sum:', sum, 'Time:' , delta color col_total.
endform.                                               "VERSION1
```

Exercise 2: Minimize the Amount of Transferred Data – Specify a SELECT List

- Open program ZABAP152_2_xx for editing (xx = the number of your group).
- Optimize the SELECT statements in form “version2”: Select only the columns needed in the subroutine.
- Run the program to see the effect of your optimization.

Solution 2: Minimize the Amount of Transferred Data – Specify a SELECT List

```

*-----*
*  FORM VERSION2
*-----*
*
*-----*
form version2.
* Version 2
  data: sflight_wa type sflight,
        sbook_wa type sbook,
        sum type sbook-loccuram value 0,
        t1 type i, t2 type i, delta(16) type p.

  get run time field t1.                                "start clock

* SELECT specific columns, specific rows
select carrid connid fldate from sflight
  into (sflight_wa-carrid, sflight_wa-connid, sflight_wa-fldate)
  where carrid = 'LH'
     and seatsmax_f = sflight~seatsocc_f.
select loccuram from sbook into sbook_wa-loccuram
  where carrid = sflight_wa-carrid
     and connid = sflight_wa-connid
     and fldate = sflight_wa-fldate
     and loccurkey = 'EUR'.
  sum = sum + sbook_wa-loccuram.
endselect.
endselect.

  get run time field t2.                                "stop clock
  delta = t2 - t1.
  skip.
  write: / 'SELECT specific columns, specific rows:' color col_heading.
  write: / 'Sum:', sum, 'Time:' , delta color col_total.
endform.                                                "VERSION2

```

Optional Exercise 2.1: Use FOR ALL ENTRIES

- Open program ZABAP152_6_xx for editing (xx = the number of your group).
- Optimize the SELECT statements in form “version2_1”: Replace the nested SELECT loops by SELECT ... INTO TABLE and SELECT ... FOR ALL ENTRIES.
- Run the program to see the effect of your optimization.

Solution 2.1: Use FOR ALL ENTRIES

```

*-----*
*  FORM VERSION2_1
*-----*
*
*-----*
form version2_1.
* Version 2_1
  data: sflight_tab type standard table of sflight,
        sbook_wa type sbook,
        sum type sbook-loccuram value 0,
        t1 type i, t2 type i, delta(16) type p.

  get run time field t1.                                "start clock

* SELECT specific rows, use FOR ALL ENTRIES
  select carrid connid fldate from sflight
     into corresponding fields of table sflight_tab
     where carrid = 'LH'
        and seatsmax_f = sflight~seatsocc_f.
  select bookid loccuram from sbook
     into (sbook_wa-bookid, sbook_wa-loccuram)
     for all entries in sflight_tab
     where carrid = sflight_tab-carrid
        and connid = sflight_tab-connid
        and fldate = sflight_tab-fldate
        and loccurkey = 'EUR'.
     sum = sum + sbook_wa-loccuram.
endselect.

  get run time field t2.                                "stop clock
  delta = t2 - t1.
  skip.
  write: / 'SELECT cols, rows, FOR ALL ENTRIES:' color col_heading.
  write: / 'Sum:', sum, 'Time:', delta color col_total.
endform.                                                "VERSION2_1

```

Exercise 3: Minimize the Amount of Transferred Data – Use Aggregate Functions

- Open program ZABAP152_3_xx for editing (xx = the number of your group).
- Optimize the SELECT statements in form “version3”: Have the database calculate the sum in the inner loop.
- Run the program to see the effect of your optimization.

Solution 3: Minimize the Amount of Transferred Data – Use Aggregate Functions

```

*-----*
*  FORM VERSION3
*-----*
*
*-----*
form version3.
* Version 3
  data: sflight_wa type sflight,
        sbook_wa type sbook,
        sum type sbook-loccuram value 0,
        t1 type i, t2 type i, delta(16) type p.

  get run time field t1.                                "start clock

* SELECT specific rows, aggregate function.
  sum = 0.
  select carrid connid fldate from sflight
         into (sflight_wa-carrid, sflight_wa-connid, sflight_wa-fldate)
         where carrid = 'LH'
              and seatsmax_f = sflight~seatsocc_f.
  select sum( loccuram ) from sbook into sbook_wa-loccuram
         where carrid = sflight_wa-carrid
              and connid = sflight_wa-connid
              and fldate = sflight_wa-fldate
              and loccurkey = 'EUR'.
  sum = sum + sbook_wa-loccuram.
endselect.

  get run time field t2.                                "stop clock
  delta = t2 - t1.
  skip.
  write: / 'SELECT specific rows, aggregate function:' color col_heading.
  write: / 'Sum:', sum, 'Time:', delta color col_total.
endform.                                                "VERSION3

```

Exercise 4: Keep the Number of Roundtrips Small – Use an Inner Join

- Open program ZABAP152_4_xx for editing (xx = the number of your group).
- Optimize the SELECT statements in form “version4”: Replace the nested SELECT statement by an inner join.
- Run the program to see the effect of your optimization.

Solution 4: Keep the Number of Roundtrips Small – Use an Inner Join

```

*-----*
*      Form  VERSION4
*-----*
* This is the fastest way to get the result!
*-----*
form version4.
* Version 4
  data: sum type sbook-loccuram value 0,
        t1 type i, t2 type i, delta(16) type p.

  get run time field t1.                                "start clock

* SELECT specific rows, aggregate function, JOIN.
  select sum( loccuram )
    from sflight as f inner join sbook as b
      on f~carrid = b~carrid and
      f~connid = b~connid and
      f~fldate = b~fldate
  into sum
  where f~carrid      = 'LH'
     and f~seatsmax_f = f~seatsocc_f
     and loccurkey    = 'EUR'.

  get run time field t2.                                "stop clock
  delta = t2 - t1.
  write: / 'specific rows, aggregate function, JOIN:' color col_heading.
  write: / 'Sum:', sum, 'Time:', delta color col_positive.
endform.                                                "VERSION4

```

Optional Exercise 4.1: Keep the Number of Roundtrips Small – FOR ALL ENTRIES again

- Open program ZABAP152_8_xx for editing (xx = the number of your group).
- Find all cities reachable from a given starting point. This can be done with a single SELECT ... FOR ALL ENTRIES statement (in a loop).
- Run the program to see whether you find all destinations.

Solution 4.1: Keep the Number of Roundtrips Small – FOR ALL ENTRIES again

```

*****
* R E A C H A B L E _ F R O M                                     *
*                                                                 *
* This form creates a list of all cities that are reachable starting *
* from a given airport.                                           *
*                                                                 *
* Parameters: IN  start: name of the starting point                *
*               OUT  ././                                         *
*                                                                 *
*****
form reachable_from
  using start type spfli-cityfrom.
  data: itab type standard table of spfli,
        wa  type spfli.
  data: visited type range of spfli-cityto with header line.

* initialise the table of already visited destinations
visited-sign = 'I'.
visited-option = 'EQ'.
visited-low = start.
append visited.

* initialise FOR ALL ENTRIES table
wa-cityto = start.
append wa to itab.

* for each loop use output of previous loop as input
do.
  select * from spfli
         into table itab
         for all entries in itab
         where cityfrom = itab-cityto
            and cityto not in visited.
         " INTO table and FAE table
         " are the same, not allowed
         " in releases <= 46D!
         " avoid endless recursion
  if sy-subrc <> 0.
    exit.
  endif.
  loop at itab into wa.
    write: / wa-cityfrom, wa-cityto, wa-carrid, wa-connid.
    visited-low = wa-cityto.
    append visited.
  endloop.
  uline.
enddo.
endform.

```

Optional Exercise 5: Keep the Number of Roundtrips Small – Use a Subquery

- Open program ZABAP152_10_xx for editing (xx = the number of your group).
- Optimize the SELECT statements in form "version3": Replace the two SELECT statements by a subquery.
- Run the program to see the effect of your optimization.

```

*-----*
*  FORM VERSION3
*-----*
*  Select all flights departing from Frankfurt in 2003 that are not
*  fully booked.
*-----*
form version3.
* Version 3
  data: sflight_tab type standard table of sflight,
        cnt type i,
        t1 type i, t2 type i, delta(16) type p.

  refresh sflight_tab.

  get run time field t1.                                "start clock

  select * from sflight as f
  into table sflight_tab
  where seatsocc < f~seatsmax
     and exists ( select * from spfli
                  where carrid = f~carrid
                    and connid = f~connid
                    and cityfrom = 'FRANKFURT' )
     and fldate between '20030101' and '20031231'.

  get run time field t2.                                "stop clock
  delta = t2 - t1.
  skip.
  write: / 'SELECT with subquery:' color col_heading.
  describe table sflight_tab lines cnt.
  write: / 'Found', cnt, 'connections.',
        / 'Time:', delta color col_positive.
endform.                                                "VERSION3

```

Optional Exercise 6: Keep the Number of Roundtrips Small – Use a Subquery

- Open program ZABAP152_12_xx for editing (xx = the number of your group).
- Optimize the SELECT statements in form "version2": Replace the two SELECT statements by one SELECT using subqueries and / or joins.
- Run the program to see the effect of your optimization.

```

*-----*
*  FORM VERSION2
*-----*
*  Find the cheapest flight(s) from FRANKFURT to NEW YORK.
*-----*
form version2.
* Version 2
  data: sflight_wa type sflight,
        t1 type i, t2 type i, delta(16) type p.

  get run time field t1.                                "start clock

  select f1~carrid f1~connid fldate price
         from sflight as f1 inner join spfli as p1
         on  f1~carrid = p1~carrid and
            f1~connid = p1~connid
         into (sflight_wa-carrid, sflight_wa-connid,
              sflight_wa-fldate, sflight_wa-price)
         where cityfrom = 'FRANKFURT'
            and  cityto  = 'NEW YORK'
            and  price =
      ( select min( price )
        from sflight as f2 inner join spfli as p2
        on  f2~carrid = p2~carrid and
           f2~connid = p2~connid
        where cityfrom = 'FRANKFURT'
           and  cityto  = 'NEW YORK'
        ).
  write: / sflight_wa-carrid, sflight_wa-connid,
         sflight_wa-fldate, sflight_wa-price.
endselect.

  get run time field t2.                                "stop clock
  delta = t2 - t1.
  write: / 'Found', sy-dbcnt, 'connections.',
         / 'Time:', delta color col_positive.
endform.                                                "VERSION2

```

Copyright 2003 SAP AG. All Rights Reserved

- No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.
- Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.
- Microsoft®, WINDOWS®, NT®, EXCEL®, Word®, PowerPoint® and SQL Server® are registered trademarks of Microsoft Corporation.
- IBM®, DB2®, DB2 Universal Database, OS/2®, Parallel Sysplex®, MVS/ESA, AIX®, S/390®, AS/400®, OS/390®, OS/400®, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere®, Netfinity®, Tivoli®, Informix and Informix® Dynamic Server™ are trademarks of IBM Corporation in USA and/or other countries.
- ORACLE® is a registered trademark of ORACLE Corporation.
- UNIX®, X/Open®, OSF/1®, and Motif® are registered trademarks of the Open Group.
- Citrix®, the Citrix logo, ICA®, Program Neighborhood®, MetaFrame®, WinFrame®, VideoFrame®, MultiWin® and other Citrix product names referenced herein are trademarks of Citrix Systems, Inc.
- HTML, DHTML, XML, XHTML are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.
- JAVA® is a registered trademark of Sun Microsystems, Inc.
- JAVASCRIPT® is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.
- MarketSet and Enterprise Buyer are jointly owned trademarks of SAP AG and Commerce One.
- SAP, SAP Logo, R/2, R/3, mySAP, mySAP.com, xApps, mySAP Business Suite, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies.

SAP assumes no responsibility for errors or omissions in these materials.