Challenge Augustus 2023

Soldier Payment Rules – part 2

A second solution with DT5GL/SQL by Jack Jansonius – 1 February 2024

Comments on this solution:
In Part 1, I had included all hourly rates in database tables, for example, for profession:

- profession rate -

professionid	rate	date_from	date_to
1	2	2015-01-01	2016-06-30
1	2.5	2016-07-01	NULL
2	1	2015-01-01	2016-06-30
2	1.6	2016-07-01	NULL
3	1	2015-01-01	2016-06-30
3	1.5	2016-07-01	NULL
4	3	2015-01-01	2016-06-30
4	4	2016-07-01	NULL

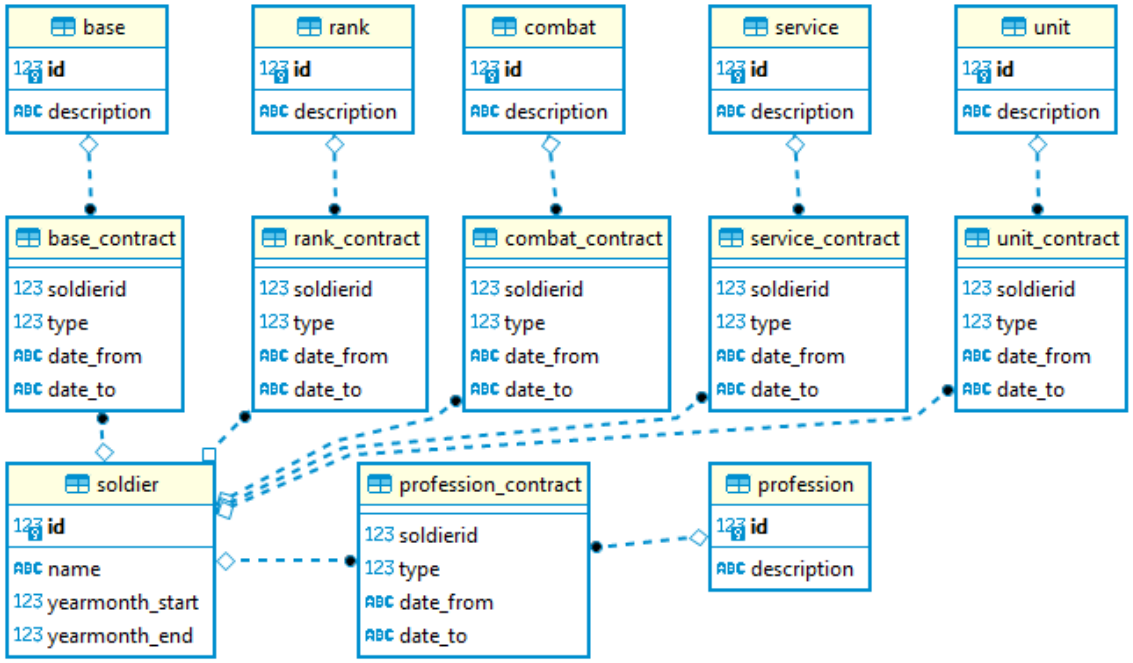
- profession -

id	description
1	fighter
2	driver
3	cook
4	officer

In this part, I determine hourly rates based on decision tables, such as:

```
rTable 5:
If:                                  | 0| 1| 2| 3| 4| 5| 6| 7| 8|
feature.profession = "fighter"       | -| Y| Y| -| -| -| -| -| -|
feature.profession = "driver"        | -| -| -| Y| Y| -| -| -| -|
feature.profession = "cook"          | -| -| -| -| -| Y| Y| -| -|
feature.profession = "officer"       | -| -| -| -| -| -| -| Y| Y|
checkdate < 20150101                 | Y| N| N| N| N| N| N| N| N|
checkdate < 20160701                 | -| Y| N| Y| N| Y| N| Y| N|
Then:
profession_rate = 0                  | X|  |  |  |  |  |  |  |  |
profession_rate = 2.00               |  | X|  |  |  |  |  |  |  |
profession_rate = 2.50               |  |  | X|  |  |  |  |  |  |
profession_rate = 1.00               |  |  |  | X|  | X|  |  |  |
profession_rate = 1.60               |  |  |  |  | X|  |  |  |  |
profession_rate = 1.50               |  |  |  |  |  |  | X|  |  |
profession_rate = 3.00               |  |  |  |  |  |  |  | X|  |
profession_rate = 4.00               |  |  |  |  |  |  |  |  | X|
# .......
```

ER-Diagram (DBeaver/SQlite)



Tables in the database:

- soldier -

id	name	yearmonth_start	yearmonth_end
1	Not Retired	201501	201812
2	Retired	201501	201512
3	Regretting Retirement	201502	201712

- service contract -

soldierid	type	date_from	date_to
1	1	2015-01-01	2015-06-30
1	2	2015-07-01	2015-11-30
1	1	2015-12-01	NULL
2	3	2015-01-01	NULL
3	1	2015-01-01	2015-06-30
3	3	2015-07-01	2015-11-30
3	2	2015-12-01	NULL

- service -

id	description
1	active
2	reserve
3	retired

- base contract -

soldierid	type	date_from	date_to
1	1	2015-01-01	NULL
3	1	2015-01-01	NULL

- base -

id	description
1	base

- rank contract -

soldierid	type	date_from	date_to
1	1	2015-01-01	2015-12-31
1	2	2016-01-01	2016-12-31
1	3	2017-01-01	NULL
3	5	2015-01-01	2015-12-31
3	4	2016-01-01	2016-12-31
3	3	2017-01-01	NULL

- rank -

id	description
1	private
2	corporal
3	sergeant
4	lieutenant
5	captain

- profession contract

soldierid	type	date_from	date_to
1	1	2015-01-01	2015-06-30
1	3	2015-07-01	2015-11-30
1	2	2015-12-01	2016-12-31
1	4	2017-01-01	NULL
3	1	2015-01-01	2015-06-30
3	2	2015-07-01	2015-11-30
3	3	2015-12-01	2016-12-31
3	4	2017-01-01	NULL

- profession -

id	description
1	fighter
2	driver
3	cook
4	officer

- unit contract

soldierid	type	date_from	date_to
1	1	2015-01-01	2015-12-31
1	1	2016-01-01	NULL
3	2	2015-01-01	2015-12-31
3	1	2016-01-01	NULL

- unit -

	id	description
1	1	HQ
2	2	paratroopers
3	3	marines
4	4	infantry

- combat contract

soldierid	type	date_from	date_to
1	2	2015-01-01	2015-03-30
1	1	2015-04-01	2015-06-30
1	2	2015-07-01	NULL
3	1	2015-01-01	2015-03-30
3	2	2015-04-01	2015-06-30
3	1	2015-07-01	NULL

- combat -

id	description
1	yes
2	no

Implementation of the decision model in DT5GL (part 2):

```
SQLite_database: "Database/Soldier Payment2.db"

# Reference day = the 15th of the month.
# Hourly rates moved from database tables (part 1) to decision tables.

Table 0:
If:                                          | 0| 1|
'Next soldier present'                       | Y| N|
Then:
NextSoldier is Selected                      | X|  |
NextSoldier is Finished                      |  | X|
# .......
# Repeat until: Finished

Proposition: 'Next soldier present'
Obtain_instance_from_database_view: soldier

Table 1:
If:                                          | 0| 1|
Next year in [firstYear-lastYear]            | Y| N|
Then:
EvalYear is Selected                         | X|  |
EvalYear is Finished                         |  | X|
# .......
# Repeat until: Finished

Table 2:
If:                                          | 0| 1| 2| 3| 4|
Next month in [firstMonth-lastMonth]         | Y| Y| Y| Y| N|
feature.service = "retired"                  | Y| Y| N| N| -|
current_pay_rate = 0                         | Y| N| -| -| -|
current_pay_rate = pay_rate_this_month       | -| -| Y| N| -|
Then:
EvalMonth is Skip                            | X|  | X|  |  |
EvalMonth is NoPayRate                       |  | X|  |  |  |
EvalMonth is NewPayRate                      |  |  |  | X|  |
EvalMonth is Finished                        |  |  |  |  | X|
# .......
# Repeat until: Finished


# Determine range of years to be checked: [firstYear-lastYear]
Attribute: firstYear                 Type: Integer
Equals: int(soldier.yearmonth_start/100)

Attribute: lastYear                  Type: Integer
Equals: int(soldier.yearmonth_end/100)

# Determine range of months to be checked within selected year: [firstMonth-
lastMonth]
Attribute: firstMonth                Type: Integer
Equals: soldier.yearmonth_start % 100 if year == firstYear else 1

Attribute: lastMonth                 Type: Integer
Equals: soldier.yearmonth_end % 100 if year == lastYear  else 12

# reference date for retrieving data from the database ("yyyy-mm-dd")
Attribute: reference_date            Type: Text
Equals: str(year) + "-" + zerofill(month,2) + "-15"

# reference date for use in conditions (yyyymmdd)
Attribute: checkdate
Equals: to_int(reference_date)
```

```
Attribute: fromdate                    Type: Text
Equals: "01/" + zerofill(month,2) + "/" + str(year)

Attribute: soldier.yearmonth_start      Type: Integer
Attribute: soldier.yearmonth_end        Type: Integer

Attribute: pay_rate_this_month  Type: Real
Equals: base_rate + rank_rate + profession_rate + service_rate + unit_rate +
combat_rate
```

rTable 3:

If:	0	1	2
feature.base = "base"	-	Y	Y
checkdate < 20150101	Y	N	N
checkdate < 20160701	-	Y	N
Then:			
base_rate = 0	X		
base_rate = 1.00		X	
base_rate = 1.25			X
#			

rTable 4:

If:	0	1	2	3	4	5	6	7	8	9	10
feature.rank = "private"	-	Y	Y	-	-	-	-	-	-	-	-
feature.rank = "corporal"	-	-	-	Y	Y	-	-	-	-	-	-
feature.rank = "sergeant"	-	-	-	-	-	Y	Y	-	-	-	-
feature.rank = "lieutenant"	-	-	-	-	-	-	-	Y	Y	-	-
feature.rank = "captain"	-	-	-	-	-	-	-	-	-	Y	Y
checkdate < 20150101	Y	N	N	N	N	N	N	N	N	N	N
checkdate < 20160701	-	Y	N	Y	N	Y	N	Y	N	Y	N
Then:											
rank_rate = 0	X										
rank_rate = 1.00		X									
rank_rate = 1.50			X								
rank_rate = 2.00				X							
rank_rate = 2.50					X						
rank_rate = 3.00						X					
rank_rate = 3.50							X				
rank_rate = 4.00								X			
rank_rate = 4.50									X		
rank_rate = 5.00										X	
rank_rate = 5.50											X
#											

rTable 5:

If:	0	1	2	3	4	5	6	7	8
feature.profession = "fighter"	-	Y	Y	-	-	-	-	-	-
feature.profession = "driver"	-	-	-	Y	Y	-	-	-	-
feature.profession = "cook"	-	-	-	-	-	Y	Y	-	-
feature.profession = "officer"	-	-	-	-	-	-	-	Y	Y
checkdate < 20150101	Y	N	N	N	N	N	N	N	N
checkdate < 20160701	-	Y	N	Y	N	Y	N	Y	N
Then:									
profession_rate = 0	X								
profession_rate = 2.00		X							
profession_rate = 2.50			X						
profession_rate = 1.00				X		X			
profession_rate = 1.60					X				
profession_rate = 1.50							X		
profession_rate = 3.00								X	
profession_rate = 4.00									X
#									

rTable 6:

If:	0	1	2	3	4	5
feature.service = "active"	-	Y	Y	-	-	-
feature.service = "reserve"	-	-	-	Y	Y	-
feature.service = "retired"	-	-	-	-	-	Y
checkdate < 20150101	Y	N	N	N	N	-
checkdate < 20160701	-	Y	N	Y	N	-
Then:						
service_rate = 0	X					X
service_rate = 2.00		X				
service_rate = 2.50			X			
service_rate = 1.00				X		
service_rate = 1.50					X	

....

rTable 7:

If:	0	1	2	3	4
feature.unit = "HQ"	-	Y	Y	-	-
OtherUnit = 1	-	-	-	Y	Y
checkdate < 20150101	Y	N	N	N	N
checkdate < 20160701	-	Y	N	Y	N
Then:					
unit_rate = 0	X				
unit_rate = 1.00		X			
unit_rate = 1.50			X		
unit_rate = 2.00				X	
unit_rate = 2.50					X

....

Attribute: OtherUnit
Equals: 1 if feature.unit in ["paratroopers", "marines", "infantry"] else 0

rTable 8:

If:	0	1	2	3
feature.combat = "yes"	-	Y	Y	-
feature.combat = "no"	-	-	-	Y
checkdate < 20150101	Y	N	N	-
checkdate < 20160701	-	Y	N	-
Then:				
combat_rate = 0	X			X
combat_rate = 5.00		X		
combat_rate = 5.50			X	

....

```
########################## Database views ##########################

Database_view: soldier
With_attributes: id, name, yearmonth_start, yearmonth_end
Query:
 SELECT id, name, yearmonth_start, yearmonth_end
  FROM soldier
 LIMIT 1 OFFSET %s
With_arguments: soldier.auto_index


Database_view: feature[1]
With_attributes: base, rank, profession, service, unit, combat
Query:
WITH input_date AS (SELECT '%s' AS refdate)[2]
SELECT b.description AS base,
       r.description AS rank,
       p.description AS profession,
       serv.description AS service,
       u.description AS unit,
       cb.description AS combat
  FROM soldier s
       LEFT JOIN base_contract bc ON s.id = bc.soldierid AND
             (SELECT refdate FROM input_date) BETWEEN bc.date_from AND
                                                     COALESCE(bc.date_to,'9999-12-31')
       LEFT JOIN base b ON bc.type = b.id
       LEFT JOIN rank_contract rc ON s.id = rc.soldierid AND
             (SELECT refdate FROM input_date) BETWEEN rc.date_from AND
                                                     COALESCE(rc.date_to,'9999-12-31')
       LEFT JOIN rank r ON rc.type = r.id
       LEFT JOIN profession_contract pc ON s.id = pc.soldierid AND
             (SELECT refdate FROM input_date) BETWEEN pc.date_from AND
                                                     COALESCE(pc.date_to,'9999-12-31')
       LEFT JOIN profession p ON pc.type = p.id
       LEFT JOIN service_contract sc ON s.id = sc.soldierid AND
             (SELECT refdate FROM input_date) BETWEEN sc.date_from AND
                                                     COALESCE(sc.date_to,'9999-12-31')
       LEFT JOIN service serv ON sc.type = serv.id
       LEFT JOIN unit_contract uc ON s.id = uc.soldierid AND
             (SELECT refdate FROM input_date) BETWEEN uc.date_from AND
                                                     COALESCE(uc.date_to,'9999-12-31')
       LEFT JOIN unit u ON uc.type = u.id
       LEFT JOIN combat_contract cc ON s.id = cc.soldierid AND
             (SELECT refdate FROM input_date) BETWEEN cc.date_from AND
                                                     COALESCE(cc.date_to,'9999-12-31')
       LEFT JOIN combat cb ON cc.type = cb.id
 WHERE s.id = %s
With_arguments: reference_date, soldier.id
```

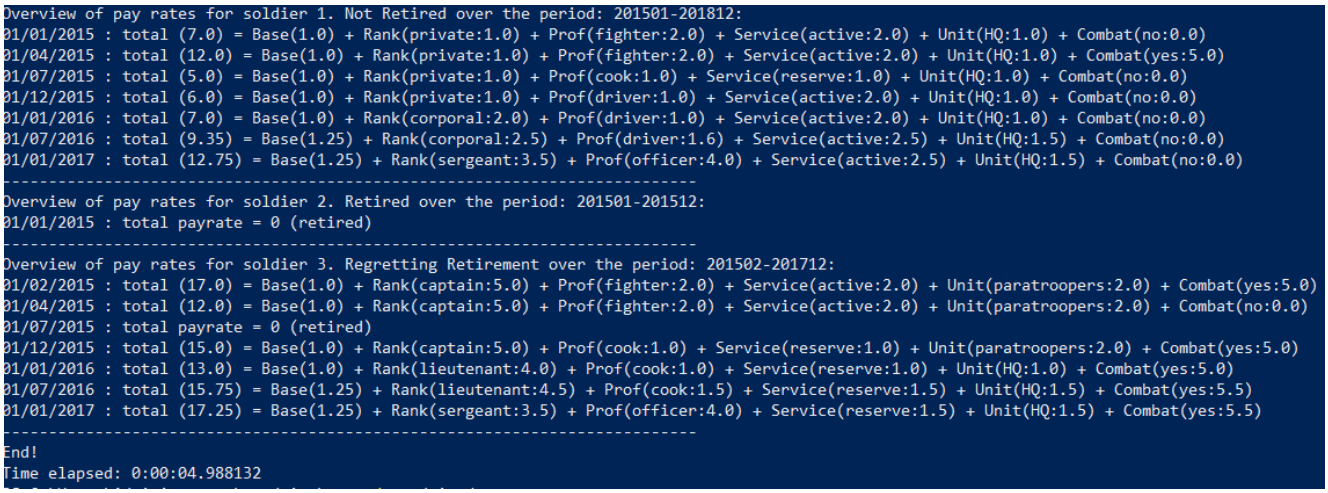
[1] Of course, the actual characteristics on the reference date can also be retrieved with 6 different views, as in the first solution.

[2] As suggested by ChatGPT for SQLite. In PostgreSQL, this can also be done with a variable: DECLARE @refdate DATE = '%s'; then any (SELECT refdate FROM input_date) in the query can be replaced by @refdate.

```
########################## GoalAttributes ############################

GoalAttribute: NextSoldier
Repeat_until:  Finished

Case: Finished
Print: "End!"

Case: Selected
Print: "Overview of pay rates for soldier %s. %s over the period: %s-%s:"
soldier.id soldier.name soldier.yearmonth_start soldier.yearmonth_end
>>: current_pay_rate = -1.00         #


GoalAttribute: EvalYear
Repeat_until:  Finished

Case: Finished
Print: "--------------------------------------------------------------------------
"

Case: Selected
Print: "#REM# -  "


GoalAttribute: EvalMonth
Repeat_until: Finished

Case: Finished
Print: "#REM# -  "

Case: Skip
Print: "#REM# -  "

Case: NoPayRate
Print: "%s : total payrate = 0 (retired) "  fromdate
>>: current_pay_rate = 0

Case: NewPayRate
Print: "%s : total (%s) = Base(%s) + Rank(%s:%s) + Prof(%s:%s) + Service(%s:%s) +
Unit(%s:%s) + Combat(%s:%s)" fromdate pay_rate_this_month base_rate feature.rank
rank_rate feature.profession profession_rate feature.service service_rate
feature.unit unit_rate feature.combat combat_rate
>>: current_pay_rate = pay_rate_this_month
```

Testrun part 2:

```
Overview of pay rates for soldier 1. Not Retired over the period: 201501-201812:
01/01/2015 : total (7.0) = Base(1.0) + Rank(private:1.0) + Prof(fighter:2.0) +
Service(active:2.0) + Unit(HQ:1.0) + Combat(no:0.0)
01/04/2015 : total (12.0) = Base(1.0) + Rank(private:1.0) + Prof(fighter:2.0) +
Service(active:2.0) + Unit(HQ:1.0) + Combat(yes:5.0)
01/07/2015 : total (5.0) = Base(1.0) + Rank(private:1.0) + Prof(cook:1.0) +
Service(reserve:1.0) + Unit(HQ:1.0) + Combat(no:0.0)
01/12/2015 : total (6.0) = Base(1.0) + Rank(private:1.0) + Prof(driver:1.0) +
Service(active:2.0) + Unit(HQ:1.0) + Combat(no:0.0)
01/01/2016 : total (7.0) = Base(1.0) + Rank(corporal:2.0) + Prof(driver:1.0) +
Service(active:2.0) + Unit(HQ:1.0) + Combat(no:0.0)
01/07/2016 : total (9.35) = Base(1.25) + Rank(corporal:2.5) + Prof(driver:1.6) +
Service(active:2.5) + Unit(HQ:1.5) + Combat(no:0.0)
01/01/2017 : total (12.75) = Base(1.25) + Rank(sergeant:3.5) + Prof(officer:4.0) +
Service(active:2.5) + Unit(HQ:1.5) + Combat(no:0.0)
------------------------------------------------------------------------
Overview of pay rates for soldier 2. Retired over the period: 201501-201512:
01/01/2015 : total payrate = 0 (retired)
------------------------------------------------------------------------
Overview of pay rates for soldier 3. Regretting Retirement over the period: 201502-
201712:
01/02/2015 : total (17.0) = Base(1.0) + Rank(captain:5.0) + Prof(fighter:2.0) +
Service(active:2.0) + Unit(paratroopers:2.0) + Combat(yes:5.0)
01/04/2015 : total (12.0) = Base(1.0) + Rank(captain:5.0) + Prof(fighter:2.0) +
Service(active:2.0) + Unit(paratroopers:2.0) + Combat(no:0.0)
01/07/2015 : total payrate = 0 (retired)
01/12/2015 : total (15.0) = Base(1.0) + Rank(captain:5.0) + Prof(cook:1.0) +
Service(reserve:1.0) + Unit(paratroopers:2.0) + Combat(yes:5.0)
01/01/2016 : total (13.0) = Base(1.0) + Rank(lieutenant:4.0) + Prof(cook:1.0) +
Service(reserve:1.0) + Unit(HQ:1.0) + Combat(yes:5.0)
01/07/2016 : total (15.75) = Base(1.25) + Rank(lieutenant:4.5) + Prof(cook:1.5) +
Service(reserve:1.5) + Unit(HQ:1.5) + Combat(yes:5.5)
01/01/2017 : total (17.25) = Base(1.25) + Rank(sergeant:3.5) + Prof(officer:4.0) +
Service(reserve:1.5) + Unit(HQ:1.5) + Combat(yes:5.5)
------------------------------------------------------------------------
End!
Time elapsed: 0:00:04.988132
```



Comment on capturing hourly rates in decision tables

The decision tables in the examples are nicely compact because a rate change only occurs on one date and, moreover, for all roles on the same date. In reality, hourly rates will be adjusted periodically much more often, and although periodic adjustments for all roles often occur on the same date, there may again be exceptions to this. In that case, a decision table for one characteristic can become very broad and cluttered.

However, it is very easy to further break down a decision table, for example by:

```
rTable 5:
If:                                 | 0| 1| 2| 3| 4| 5| 6| 7| 8|
feature.profession = "fighter"      | -| Y| Y| -| -| -| -| -| -|
feature.profession = "driver"       | -| -| -| Y| Y| -| -| -| -|
feature.profession = "cook"         | -| -| -| -| -| Y| Y| -| -|
feature.profession = "officer"      | -| -| -| -| -| -| -| Y| Y|
checkdate < 20150101                | Y| N| N| N| N| N| N| N| N|
checkdate < 20160701                | -| Y| N| Y| N| Y| N| Y| N|
Then:
profession_rate = 0                 | X|  |  |  |  |  |  |  |  |
profession_rate = 2.00              |  | X|  |  |  |  |  |  |  |
profession_rate = 2.50              |  |  | X|  |  |  |  |  |  |
profession_rate = 1.00              |  |  |  | X|  | X|  |  |  |
profession_rate = 1.60              |  |  |  |  | X|  |  |  |  |
profession_rate = 1.50              |  |  |  |  |  |  | X|  |  |
profession_rate = 3.00              |  |  |  |  |  |  |  | X|  |
profession_rate = 4.00              |  |  |  |  |  |  |  |  | X|
# .......
```

converting to 5 consecutive tables:

```
rTable 5a:
If:                                 | 0|
checkdate < 20150101                | Y|
Then:
profession_rate = 0                 | X|
# .......

rTable 5b:
If:                                 | 0| 1|
feature.profession = "fighter"      | Y| Y|
checkdate < 20150101                | N| N|
checkdate < 20160701                | Y| N|
Then:
profession_rate = 2.00              | X|  |
profession_rate = 2.50              |  | X|
# .......

rTable 5c:
If:                                 | 0| 1|
feature.profession = "driver"       | Y| Y|
checkdate < 20150101                | N| N|
checkdate < 20160701                | Y| N|
Then:
profession_rate = 1.00              | X|  |
profession_rate = 1.60              |  | X|
# .......
```

rTable 5d:

If:	0	1
feature.profession = "cook"	Y	Y
checkdate < 20150101	N	N
checkdate < 20160701	Y	N
Then:		
profession_rate = 1.00	X	
profession_rate = 1.50		X

....

rTable 5e:

If:	0	1
feature.profession = "officer"	Y	Y
checkdate < 20150101	N	N
checkdate < 20160701	Y	N
Then:		
profession_rate = 3.00	X	
profession_rate = 4.00		X

....