Parameter:

todaysdate : normalerweise “current date”,ausser wenn Tests gemacht werden sollen mit einem bestimmten Buchungsdatum

itemroomkey : Zimmer Schlüssel

itemdatebegin : Anfangsdatum der Leistung, oder der Peridoe in der ein Special gesucht warden soll. In einer duration kann es mehrere Specials geben und dann muss am Ende der vorherigen Special-Periode erneut mit der Suche angefangen werden.

itemdateend :

itemduration : duration vom itemdatebegin bis zum letzten Tag der Leistung

alreadyusedrevolvinggroups : ('1')

SQL für die Initial-Suche nach den vielleicht relevanten Specials :

select   \*   
from     specialoffer so   
where    so.roomkey = itemroomkey   
and      ( (   
                           itemdatebegin between so.datefrom and so.dateto   
                  and      itemdateend between so.datefrom and so.dateto )   
         or       (   
                           so.datefrom between itemdatebegin and itemdateend   
                  and      so.dateto between itemdatebegin and itemdateend )   
         or       (   
                           itemdatebegin between so.datefrom and so.dateto   
                  and      itemdateend > so.dateto )   
         or       (   
                           itemdateend between so.datefrom and so.dateto   
                  and      itemdatebegin < so.datefrom ) )   
and      ( (   
                           so.ruletype = 'Date'   
                  and      todaysdate between so.datebeforedeparturefrom and so.datebeforedepartureto )  
         or       (   
                           so.ruletype = 'Nr'   
                  and      todaysdate between itemdatebegin – so.daysbeforedeparturefrom days and itemdatebegin – so.daysbeforedepartureto days )  
         or       (   
                           so.ruletype = 'Always' )   
and      so.days <= itemduration

and      so.childadultnr = 0

and      so.childchildnr = 0   
and      so.revolvinggroup not in alreadyusedrevolvinggroups   
order by so.days desc,   
         so.todaybase desc,   
         so.datefrom asc,   
         so.dateto desc

Das SQL oben liefert für „normale“ Specials folgende Specials für einen bestimmten Buchungszeitraum, definiert durch itemdatebegin und itemdateend, zurück:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | Buchungszeitraum | |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  | Special1 | |  |  |  |  |  |  |
|  |  |  |  |  |  |  | Special3 |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  | Special4 |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  | Special2 | |  |  |  |

Wenn keine Rows zurückgeliefert werden, dann kann ich mit der Berechnung aufhören.

Es muss eine Liste geben mit allen Tagespreisen für einen bestimmten Pax für dem die Special-Preis-Berechnung erfolgen soll. Jeder Pax muss einzeln berechnet werden.

Wenn ein Pax (Adult oder Child) keine Tagespreise hat, darf sowieso kein Preis berechnet werden. Es gibt Hotels die haben nicht im ganzen Jahr offen. Wenn die Saison zu Ende ist kann ich nicht einfach so tun als wäre es dennoch möglich Preise zu berechnen. Es ist also sichergestellt, dass jeder preis-relevanter Pax auch einen Tagespreis hat und es keine Lücke in den Preisen gibt. Diese Tagespreisliste muss aufsteigend sortiert werden.

Jetzt muss es einen Loop durch jeden Tagespreis geben, angefangen vom kleinsten Tag.

Erster Tag:

Suche nach dem am besten passenden Special.

Loop durch jeden Special aus der Suche oben, von dem ersten bis zum letzten, oder einer gefunden wird der passt. Die Specials wurden oben ja nach bestimmten Regel sortiert, also sind die Regeln welche am besten passen könnten oben und werden zuerst angeschaut.

Wichtig: falls ein Special schon genutzt wurde, dürfen keine Specials der RevolvingGroup ein weiteres Mal genutzt werden.

Wenn itemdateend < itemdatebegin + so.days dann passt diese Leistung nicht mehr in die verfügbaren Tage um diesen Special zu nutzen und der Special kann ignoriert werden. Es muss dann mit dem nächsten Special weitergemacht werden.

Wenn grundsätzlich, auf Basis der so.days der Special passen könnte, müssen weitere Tests erfolgen:

Wenn startdaterelevant=1 und enddaterelevant=0 , und itemdatebegin < so.datefrom oder itemdatebegin > so.dateto dann liegt das Anfangsdatum der Leistung ausserhalb der der Special-Saison und der Special darf nicht zur Anwendung kommen. Es muss dann mit dem nächsten Special weitergemacht werden.

Wenn startdaterelevant=0 und enddaterelevant=1 , und itemdateend < so.datefrom oder itemdateend > so.dateto dann liegt das Anfangsdatum der Leistung ausserhalb der der Special-Saison und der Special darf nicht zur Anwendung kommen. Es muss dann mit dem nächsten Special weitergemacht werden.

Wenn startdaterelevant=1 und enddaterelevant=1 , und itemdateend < so.datefrom oder itemdateend > so.dateto oder itemdatebegin < so.datefrom oder itemdatebegin > so.dateto dann liegen das Anfangsdatum und Enddatum der Leistung ausserhalb der der Special-Saison und der Special darf nicht zur Anwendung kommen. Es muss dann mit dem nächsten Special weitergemacht werden.

Wenn wir so weit gekommen sind, dann ist dies ein „normaler“ Special, aber es gibt immer noch Regeln die getestet werden müssen um zu bestimmen ob der aktuell überprüfte Special zur Anwendung kommen darf oder nicht.

Wenn itemdatebegin < so.datefrom dann liegt das Anfangsdatum der Leistung ausserhalb der Saison und der Special kann hier nicht zur Anwendung kommen und der nächste Special muss angeschaut werden ob dieser passen könnte.

Wenn itemdatebegin > so.dateto dann gibt es eine Saisonüberschneidung. Im Falle einer Saisonüberschneidung müssen zwei Tests gemacht werden:

1. Wenn itemdateend > so.lastspoffenddate kann der Special ignoriert werden, weil die nächste Saison nicht ausreicht für den Special. Es muss mit dem nächsten Special weitergemacht werden.
2. Wenn itemdatebegin > so.dateto dann ist diese Saison des aktuell getesteten Specials schon zu Ende und der Special kann ignoriert werden.

Wir haben die Regeln unten als Pseudo-Code dokumentiert.

If so.startdaterelevant=1&so.enddaterelevant=0

If so.datefrom <= itemdatebegin

& so.dateto >= itemdatebegin

return true

Else

return false

End If

Else If so.enddaterelevant=1&so.startdaterelevant=0

If so.datefrom <= itemdateend

& so.dateto >= itemdateend

return true

Else

return false

End If

Else If so.startdaterelevant=1&so.enddaterelevant=1

If so.datefrom <= itemdatebegin

& so.dateto >= itemdatebegin

& so.datefrom <= itemdateend

& so.dateto >= itemdateend

return true

Else

return false

End If

Else

If itemdatebegin < so.datefrom

return false

End If

If itemdateend > so.dateto

If so.dateto >so.lastspoffenddate

return false

End If

If itemdatebegin > so.dateto

return false

End If

End If

End If

return true

Wenn wir so weit sind, dann könnte das Special passen. Bei einer Saisonüberschneidung gibt es weitere Regeln, das versucht werden muss das kleinste Special zu finden der am wenigsten in die neue Saison überschneidet.

Das Special welches aktuell getestet wird hat eine so.days. Auf Basis des Anfangsdatums der Leistung könnte auf Basis der so.days der Special in die neue Saison reichen oder nicht.

Wenn der Special nicht in die neue Saison überschneidet, dann gibt es zwei Fälle.

1. Es könnte sein, dass vorher ein anderes grösseres Special, mit einem grösseren Wert für so.days, gefunden wurde der aber in die nächste Saison reinreicht. Wenn dem **nicht** so war, dann muss dieses Special zur Anwendung kommen.
2. Es könnte sein, dass vorher ein anderes grösseres Special, mit einem grösseren Wert für so.days, gefunden wurde der aber in die nächste Saison reinreicht. **Wenn dem so war**, dann gilt die Regel, dass wenn itemdatebegin + so.days = so.dateend, dann muss ich diesen Special verwenden weil dieser genau in die alte Saison passt.
3. Der übrige Fall ist, dass itemdatebegin + so.days < so.dateend. Dieser Fall bedeutet das der vorher gefundene Special welcher Saisonübergreifend ist genutzt werden darf. Es hat sicher herausgestellt, dass die Specials besser funktionieren wenn die grösseren Regeln zuerst angewendet werden, auch wenn diese in die nächste Saison übergreifen. Wenn zuerst immer nur die kleinen Specials genutzt werden welche noch in die alte Saison passen könnten, kommt es öfter vor, dass der Kunde nicht die korrekten Specials bekommt, daher dieser Kompromiss.

Wenn im vorherigen Schnitt kein Special festgestellt wurde, könnte es sein, dass es eine Saisonüberschneidung gibt, aber der Special ist als startdaterelevant=1 und enddaterelevant=0 definiert. In diesem Fall kann ich das Special sofort verwenden.

Wenn immer noch kein Special ermittelt wurde, handelt es sich wahrscheinlich um einen Special der in die nächste Saison übergreift. Ist dies der Fall, dann kann ich nicht mit der Suche aufhören. Es gilt nämlich die Regel, das der kleinste Special zur Anwendung kommen soll der am wenigsten in die neue Saison ragt. Es muss also weiter geschaut werden nach einem Special der vielleicht weniger in diese neue Saison überschneidet, ich muss mir aber den aktuellen Special merken, es könnte ja sein, dass dieser der Special ist der am wenigsten in die neue Saison ragt.

Hier Pseudo-Code zu den Regeln oben:

If itemdatabegin + so.days days <= so.dateend

If SaveSpecialListLineNr > 0

If itemdatabegin + so.days days = so.dateend

Calculate SaveSpecialListLineNr as CurrentSpecialListLineNr

Break to end of loop

End If

Else

Calculate SaveSpecialListLineNr as CurrentSpecialListLineNr

Break to end of loop

End If

Else If (

itemdatabegin + so.days days > so.dateend

& so.startdaterelevant = 1

& so.enddaterelevant = 0 )

Calculate SaveSpecialListLineNr as CurrentSpecialListLineNr

Break to end of loop

Else

If so.dateend <= so.lastspoffenddate

Calculate SaveSpecialListLineNr as CurrentSpecialListLineNr

End If

End If

return SaveSpecialListLineNr

Wenn jetzt ein passendes Special gefunden wurde, dann muss dieser auf Basis des DayString die für das Special relevanten Tage in Gruppen aufteilen.

Beispiel: wenn Days = 8 und der DayString ist „4,4“, dann müssen die ersten 8 Tage aus der Tagespreis-Liste genommen werden und in zwei Gruppen mit jeweils 4 Elemente aufgeteilt werden.

1JAN + 2JAN + 3JAN + 4JAN

5JAN + 6JAN + 7JAN + 8JAN

Jede Gruppe wird einzeln angeschaut und auf Basis vom Rule (MinAmount, MaxAmount, MinDays, MaxDays, Average) nach dessen Preise oder Tage, oder es muss ein Average gerechnet werden.

* 1JAN = 101
* 2JAN = 100
* 3JAN=103
* 4JAN=104
* 5JAN = 106
* 6JAN = 105
* 7JAN=108
* 8JAN=109

Bei MinDays würde die erste Gruppe nach Tagen aufsteigend sortiert werden und der erste Tag der Liste 1JAN wäre der Special-Tag.

Bei MaxDays würde die erste Gruppe nach Tagen absteigend sortiert werden und der erste Tag der Liste 4JAN wäre der Special-Tag.

Bei MinAmount würde die erste Gruppe nach Preis aufsteigend sortiert werden und der erste Tag der Liste 2JAN wäre der Special-Tag.

Bei MaxAmount würde die erste Gruppe nach Tagen absteigend sortiert werden und der erste Tag der Liste 4JAN wäre der Special-Tag.

Bei Average müssten die Preise der ersten Gruppe 1JAN bis 4JAN zusammengerechnet werden und der Durchschnitt errechnet werden: (101+100+103+104) / 4 = 102

Dann muss die nächste Gruppe in Angriff genommen werden.

Ein Stay kann länger sein als dieses erste Special, also muss die Special-Preis-Berechnung so lange weitergeführt werden bis ein Finden von Specials ausgeschlossen werden kann.

Es wird ein neues Datum itemstartdate gerechnet, die duration entsprechend des vorher gefundenen Specials angepasst, und die Suche fängt wieder von vorne an. Bei einem Stay von 20 Nächten, wenn ich ein Special für 8 Nächte gefunden habe, dann gibt es immer noch 12 Nächte für die ein Special ermittelt und angerechnet werden könnte. Ich fange als dann am Tag 9 an, mit einer duration von 12 Tagen und suche nach dem nächsten Special.

Wichtig: wurde eine Regel aus einer RevolvingGroup einmal schon genutzt, darf in der weiteren Suche diese RevolvingGroup nicht mehr zur Anwendung kommen.

Kinder-Specials: diese kommen selten vor. Wenn ja, dann ist die Berechnung analog den für die Erwachsenen. Der einzige Unterschied ist, dass beim ersten Kind nach

Child="1"

ChildAdultNr="2"

ChildChildNr="1"

nach Specials gesucht werden muss, beim zweiten Kind, falls vorhanden, nach

Child="1"

ChildAdultNr="2"

ChildChildNr="2"

Die ChildAdultNr entspricht der Norm-Belegung des Zimmers aus dem Hotel-Node <NormalOccupancy>2</NormalOccupancy>.

Falls es ein drittes Kind gäbe welches im Zimmer gebucht werden könnte, würde nach

Child="1"

ChildAdultNr="2"

ChildChildNr="3"

gesucht werden.

Hier ein Beispiel von Kinder-Specials:

<SpecialOffers>

<!-- SO for child 1 when 2 adults -->

<SpecialOffer Baby="0"

Days="6"

Type="EndDays"

Child="1"

DateTo="2016-12-31"

DescId="5"

DateFrom="2016-01-01"

RuleType="Always"

AddAmount="0.00"

DayString="6"

PayNights="4"

ToDayBase="6"

FromDayBase="6"

ChildAdultNr="2"

ChildChildNr="1"

RevolvingGroup="1"

EndDateRelevant="0"

LastSpOffEndDate="2016-12-31"

StartDateRelevant="0"

DaysBeforeDepartureTo="0"

DaysBeforeDepartureFrom="0" />

<!-- SO for child 2 when 2 adults -->

<SpecialOffer Baby="0"

Days="3"

Type="EndDays"

Child="1"

DateTo="2016-12-31"

DescId="8"

DateFrom="2016-01-01"

RuleType="Always"

AddAmount="0.00"

DayString="3"

PayNights="2"

ToDayBase="3"

FromDayBase="3"

ChildAdultNr="2"

ChildChildNr="1"

RevolvingGroup="4"

EndDateRelevant="0"

LastSpOffEndDate="2016-12-31"

StartDateRelevant="0"

DaysBeforeDepartureTo="0"

DaysBeforeDepartureFrom="0" />

<!-- SO for normal occupancy i.e. adults -->

<SpecialOffer Baby="0"

Days="3"

Type="EndDays"

Child="0"

DateTo="2016-12-31"

DescId="9"

DateFrom="2016-01-01"

RuleType="Always"

AddAmount="0.00"

DayString="3"

PayNights="2"

ToDayBase="3"

FromDayBase="3"

ChildAdultNr="0"

ChildChildNr="0"

RevolvingGroup="5"

EndDateRelevant="0"

LastSpOffEndDate="2016-12-31"

StartDateRelevant="0"

DaysBeforeDepartureTo="0"

DaysBeforeDepartureFrom="0" />

</SpecialOffers>